F 1

```
   CCCCCCCCCCCC     000000000     PPPPPPPPPPPP     YYY         YYY
   CCCCCCCCCCCC     000000000     PPPPPPPPPPPP     YYY         YYY
   CCCCCCCCCCCC     000000000     PPPPPPPPPPPP     YYY         YYY
CCC              000         000  PPP         PPP  YYY         YYY
CCC              000         000  PPP         PPP  YYY         YYY
CCC              000         000  PPP         PPP  YYY         YYY
CCC              000         000  PPP         PPP     YYY   YYY
CCC              000         000  PPP         PPP     YYY   YYY
CCC              000         000  PPP         PPP     YYY   YYY
CCC              000         000  PPPPPPPPPPPP           YYY
CCC              000         000  PPPPPPPPPPPP           YYY
CCC              000         000  PPPPPPPPPPPP           YYY
CCC              000         000  PPP                    YYY
CCC              000         000  PPP                    YYY
CCC              000         000  PPP                    YYY
CCC              000         000  PPP                    YYY
CCC              000         000  PPP                    YYY
CCC              000         000  PPP                    YYY
   CCCCCCCCCCCC     000000000     PPP                    YYY
   CCCCCCCCCCCC     000000000     PPP                    YYY
   CCCCCCCCCCCC     000000000     PPP                    YYY
```

```
**FILE**ID**COPYMAIN




  CCCCCCCC    OOOOOO    PPPPPPPP    YY      YY  MM      MM    AAAAAA      IIIIII    NN      NN
  CCCCCCCC    OOOOOO    PPPPPPPP    YY      YY  MM      MM    AAAAAA      IIIIII    NN      NN
CC          OO      OO  PP      PP  YY      YY  MMMM  MMMM  AA      AA      II      NN      NN
CC          OO      OO  PP      PP  YY      YY  MMMM  MMMM  AA      AA      II      NN      NN
CC          OO      OO  PP      PP    YY  YY    MM  MM  MM  AA      AA      II      NNNN    NN
CC          OO      OO  PP      PP    YY  YY    MM  MM  MM  AA      AA      II      NNNN    NN
CC          OO      OO  PPPPPPPP        YY      MM      MM  AA      AA      II      NN  NN  NN
CC          OO      OO  PPPPPPPP        YY      MM      MM  AA      AA      II      NN  NN  NN
CC          OO      OO  PP              YY      MM      MM  AAAAAAAAAA      II      NN    NNNN
CC          OO      OO  PP              YY      MM      MM  AAAAAAAAAA      II      NN    NNNN
CC          OO      OO  PP              YY      MM      MM  AA      AA      II      NN      NN      ....
CC          OO      OO  PP              YY      MM      MM  AA      AA      II      NN      NN      ....
  CCCCCCCC    OOOOOO    PP              YY      MM      MM  AA      AA    IIIIII    NN      NN      ....
  CCCCCCCC    OOOOOO    PP              YY      MM      MM  AA      AA    IIIIII    NN      NN      ....

LL            IIIIII      SSSSSSSS
LL            IIIIII      SSSSSSSS
LL              II      SS
LL              II      SS
LL              II      SS
LL              II      SS
LL              II        SSSSSS
LL              II        SSSSSS
LL              II              SS
LL              II              SS
LL              II              SS
LL              II              SS
LLLLLLLLLL    IIIIII    SSSSSSSS
LLLLLLLLLL    IIIIII    SSSSSSSS
```

```
0 MODULE COPYMAIN (IDENT = 'V04-000',
0                   MAIN = COPY$COPY
0                   ) =
1 BEGIN
1
1 !****************************************************************************
1 !*                                                                          *
1 !*  COPYRIGHT (c) 1978, 1980, 1982, 1984 BY                                 *
1 !*  DIGITAL EQUIPMENT CORPORATION, MAYNARD, MASSACHUSETTS.                  *
1 !*  ALL RIGHTS RESERVED.                                                    *
1 !*                                                                          *
1 !*  THIS SOFTWARE IS FURNISHED UNDER A LICENSE AND MAY BE USED AND COPIED   *
1 !*  ONLY IN  ACCORDANCE WITH  THE  TERMS  OF  SUCH  LICENSE  AND WITH THE   *
1 !*  INCLUSION OF THE ABOVE COPYRIGHT NOTICE. THIS SOFTWARE OR  ANY  OTHER   *
1 !*  COPIES THEREOF MAY NOT BE PROVIDED OR OTHERWISE MADE AVAILABLE TO ANY   *
1 !*  OTHER PERSON.  NO TITLE TO AND OWNERSHIP OF  THE  SOFTWARE IS  HEREBY   *
1 !*  TRANSFERRED.                                                            *
1 !*                                                                          *
1 !*  THE INFORMATION IN THIS SOFTWARE IS  SUBJECT TO CHANGE WITHOUT NOTICE   *
1 !*  AND  SHOULD  NOT  BE  CONSTRUED AS  A COMMITMENT BY DIGITAL EQUIPMENT   *
1 !*  CORPORATION.                                                            *
1 !*                                                                          *
1 !*  DIGITAL ASSUMES NO RESPONSIBILITY FOR THE USE  OR  RELIABILITY OF ITS   *
1 !*  SOFTWARE ON EQUIPMENT WHICH IS NOT SUPPLIED BY DIGITAL.                 *
1 !*                                                                          *
1 !*                                                                          *
1 !****************************************************************************
1 !
1
1 !++
1 ! FACILITY:  COPY
1 !
1 ! ABSTRACT:
1 !
1 !       This utility program creates a copy of one or more user-specified
1 !       files.  Two or more files may optionally be concatenated to
1 !       create a single output file.
1 !
1 ! ENVIRONMENT:
1 !
1 ! AUTHOR:  Ward Clark,  CREATION DATE:  19 August 1977
1 !
1 ! Modified by:
1 !
1 !       V03-014 TSK0015         Tamar Krichevsky        26-Jul-1984
1 !               Use the constant 32 for the multi-block count, instead of
1 !               the system multi-block count.
1 !
1 !       V03-013 TSK0014         Tamar Krichevsky        9-jun-1984
1 !               Avoid an access violation by have BYPASS_CONCAT return a value.
1 !               If this value is true, then stop processing.  If it is false,
1 !               then continue copying files.
1 !
1 !       V03-012 TSK0013         Tamar Krichevsky        8-May-1984
1 !               Rearrange the calls to CLI$GET_VALUE and LIB$FIND_FILE so that
1 !               a command such as COPY a.a,a.a,a.a,a.a NL: will copy every file,
1 !               instead of every other file.
```

```
!
!       V03-011 TSK0012         Tamar Krichevsky        25-Apr-1984
!               Add a check, after trying to open the output file, to be
!               sure that if the current operation is an APPEND and the output
!               file was not found, then processing should stop.  No use
!               appending to a non-existant file.
!
!       V03-010 TSK0011         Tamar Krichevsky        17-Mar-1984
!               Add a missing ".", so that the correct files are opened when
!               the input file has a wildcard in it's specification.  Copy the
!               resultant file file name from LIB$FIND_FILE into the input
!               file's NAM block and IN_NAME_DESC.  Otherwise, the confirm
!               prompt, log messages and error reporting would use the wrong
!               information.
!
!       V03-009 TSK0010         Tamar Krichevsky        27-Feb-1984
!               Replace COPY's scheme for allocating I/O buffer pool (The I/O
!               buffer pool is area in which COPY maintains its user buffers
!               for RMS calls.)  The old scheme allocated virtual memory for
!               the I/O buffer pool based on the processes working set size.
!               The new scheme alloates enough virtual memory to hold the
!               largest record or block transfer instead.
!
!               Convert input file parse and searching to LIB$FIND_FILE.
!
!       V03-008 TSK0009         Tamar Krichevsky        15-Feb-1984
!               Fix RMS_SETUP so that the incompatible attributes message is
!               not issued when the input or the output device is network.
!
!       V03-007 TSK0008         Tamar Krichevsky        3-Oct-1983
!               Fix RMS_SETUP so that the incompatible attributes message is
!               not issued when the input device is a unit record device.
!               The input and output devices have to be the same kind of devices
!               and be file structured before the information in the file header
!               can be compared.
!
!       V03-006 TSK0007         Tamar Krichevsky        6-Sep-1983
!               Fix an Access violation introduced in V30-005.  This time
!               wild card copy operations didn't work.
!
!       V03-005 TSK0006         Tamar Krichevsky        1-Sep-1983
!               Fix access violation introduced in V30-004.  Append operations
!               didn't work.
!
!       V03-004 TSK0005         Tamar Krichevsky       29-Aug-1983
!               Modify how the output file's XAB chain is reinitialized at the
!               end of COPY$COPY.  This change has been made so that COPY
!               adheres to the new philosophy about the propogation of
!               file protection and revision dates.
!
!       V03-003 TSK0004         Tamar Krichevsky       23-Jan-1983
!               Replace the command language interface with the the new CLI.
!
!               Add COPY$CHECK_FILE_FOR_MATCH routine which calls LIB$QUAL_FILE_MATCH
!               to see if the input file should be copuied to the output file.
!
!       V03-003 TSK0003         Tamar Krichevsky       29-Mar-1982
!               Allow /NOTRUNCATE to work for non-contiguous sequential files
!               by correcting the IF statement in COPY$CALC_ALQ which decides
!               if the output file will be truncated or the same size as
!               the input file.  Previously, non-contiguous sequential files
!               were always being truncated, even if /NOTRUNCATE was specified.
!               Now, if /NOTRUNCATE is given, the allocation of the input file
!               is used for the output file.
!
!
!       V03-002 TSK0002         Tamar Krichevsky        22-Mar-1982
!               Correct logic in IF statement which forces record mode I/O
!               in RMS_SETUP.  Record mode copies to a foreign disk were being
!               attempted instead of block mode.
!
!       V03-001 TSK0001         Tamar Krichevsky        16-Mar-1982
!               Force record mode operations if input and output devices are
!               both magtape and one is ANSI while the other is mounted
!               foreign.
!
!       V021    WMC032          Wayne Cardoza   22-Dec-1981
!               Don't allow copy of a directory as a file.
!               Let the [] be displayed in mag tape log messages.
!
!       V020    WMC026          Wayne Cardoza   10-Dec-1981
!               Fix incorrect ordering of PARSE.
!               Fix log messages for network devices.
!
!       V019    WMC003          Wayne Cardoza   17-Nov-1981
!               Quit when operator aborts a mount request.
!
!       V018    WMC002          Wayne Cardoza   02-Nov-1981
!               Don't try to create directories on record devices.
!               Make sure directory created in correct directory.
!               Don't print directory name for non-directory devices.
!
!       V017    TMH0017         Tim Halvorsen   06-Sep-1981
!               Do not issue "N files created" if the number of files
!               created is only one.
!
!       X0016   KRM0007         Karl Malik      11-Feb-1981
!               Modified COPY$COPY to not attempt to create a directory
!               when the output is a network device. Instead, issue a
!               MSG$_NOTCREDIR (new) warning message and continue.
!
!       X0015   KRM0005         Karl Malik      14-Jan-1981
!               Init the block_count and record_count in CREATE_DIR so as
!               not to use the previous value. Also, modified REPORT_NAMES
!               to issue a "created" message when a subdirectory is created
!               (rather than a "copied" message).
!
!       X0014   LMK0001         Len Kawell      27-Mar-1980
!               Correct computation of USZ and MBC for record mode.
!
!       X0013   TMH0012         Tim Halvorsen   31-Jan-1980
!               Do not use LRL as the USZ for record mode I/O as the LRL
!               can sometimes be incorrect when appending files together
!               with differing LRL's.  COPY should be fixed sometime in
!                       the future to make the LRL on a concatenated file correct.
!
!       X0012   JAK0012         J. Krycka       07-Dec-1979
!               Set ASY bit in ROP after $CONNECT when doing block I/O to
!               avoid having to issue a $WAIT after the connect. This is
!               necessary for network block I/O because a network $CONNECT
!               actually causes DAP messages to be exchanged and thus does not
!               complete immediately.
!
!       X00011  TMH0011         T. Halvorsen    19-Dec-1979
!               Do not create a directory on the output side for magtapes.
!
!       X00010  TMH0010         T. Halvorsen    17-Nov-1979
!               Add GLOBAL ROUTINE msg_number from its own module to
!               this module to avoid conflict with require file of the
!               same name in the update procedure.
!               It had one modification:
!                               T. Halvorsen    15-Nov-1979
!               Do not add in COPY/APPEND facility unless high-order
!               word is non-zero.
!
!       X00009  TMH0009         T. Halvorsen    24-Oct-1979
!               If input file is a directory file, then either create
!               a directory on the output side or do nothing depending
!               on whether the directory already exists or not.
!
!       X00008  T. Halvorsen    16-Aug-1979
!               Move fixed_overhead to here from COPY.REQ and increase
!               it by another 10 to avoid copy from magtape wsl problems
!
!       X00007  T. Halvorsen    30-Jul-1979
!               Make RMS_SETUP fill the UBF/USZ fields for all device types
!               due to a change in RMS which causes move mode to always be
!               used (locate mode had some timing windows).
!
!       X00006  T. Halvorsen    21-Jul-1979
!               Remove 60 second timeout from input RAB
!
!       X00005  T. Halvorsen    14-Jul-1979
!               Detect insufficient working set size to avoid "internal logic
!               error" message when allocating negative amount of storage.
!
!       X00004  JAK0004         J. Krycka       16-Mar-1978     15:00
!               To support file append over the network, omit 'incompatible
!               attributes' check if NET bit is set.
!
!       X00003  JAK0003         J. Krycka       16-Mar-1978     14:30
!               To support copy of files in VFC format over the network,
!               put RHB address in both input and output RABs if NET bit is set.
!
!
! 01    18-04-78        C. Peters       Change INCLUDE file declarations to suit VMS native compiles.
!                                       Remove SHR$_HASHCONCAT, SHR$_INCOMPAT literals.
! 02    18-04-78        C. Peters       Change COPY to reflect modified behavior.
!       Include COPY.REQ. Delete LITERAL definitions for general use, status flags. Delete
!               macro definitions for commonly used status flags.
!       Rename COPY_STATUS to COPY$CLI_STATUS.
```

```
1 !        Don't include RMSMAC.L32, STARDE.L32. Include STARLET.L32 from SYS$LIBRARY.
1 !                Delete external literal declarations of RMS status codes. They are in STARLET.L32 too.
1 !        Delete GLOBAL variable COPY$CLI_STATUS. Put it in a new module, COPYGBL.B32.
1 !        Instead of calling GET_OUTFILE, call COPY$GET_OUTFIL, in COPYSPECS.
1 !        Delete GET_OUTFILE.
1 !        Instead of calling GET_INFILE, call COPY$GET_INFILE, in COPYSPECS.B32.
1 !        Delete GET_INFILE from this module.
1 !        Instead of calling OPEN_INFILE, call COPY$OPN_INFILE, in COPYSPECS.
1 !        Delete OPEN_INFILE.
1 !        Rename IN_OPEN_ERROR to COPY$INOPN_ERR; OUT_OPEN_ERROR to COPY$OUTOPN_ERR;
1 !                CLOSE_OUTFILE to COPY$CLOSE_OUTF.
1 !        Instead of calling OPEN_OUTFILE, call COPY$OPN_OUTFIL, in COPYSPECS.
1 !        Rename OUT_CLOSE_ERROR to COPY$OCLOSE_ERR.
1 !        Remove declaration for STS$K_INFO. Put this in COPY.REQ.
1 !        Remove declaration for VMSMAC.L32, put it in COPY.REQ.
1 !        Delete routine OPEN_OUTFILE. This routine is replaced by COPY$OPN_OUTFIL, in COPYSPECS.
1 !        Rename CALCULATE_ALQ to COPY$CALC_ALQ and make it a global routine.
1 !        Rename MESSAGE_NUMBER to COPY$MSG_NUMBER and make it a global routine.
1 !        Rename CLI_RESULT to COPY$CLI_RESULT. Declare it a global in COPYGBL.
1 !        In main routine, close output file is flag MULTIPLE_OUTPUT is set, instead of testing
1 !                for the CONCAT_FOLLOWS flag being not set.
1 !        Move setting of CONCAT_QUAL and NOCONCAT_QUAL into the routine GET_CMD_QUAL.
1 !        Move OUTFILE_OPEN and APPEND_COMMAND bits into COPY$SEM_STATUS from COPY$CLI_STATUS.
1 !        Remove RMS declarations for input file descriptions to file called FILINPUT.B32.
1 !        Remove RMS declarations for output file descriptions to file called FILOUTPUT.B32.
1 !        Rename PARSE_INFILE to COPY$PARS_INFIL.
1 !        Move PUT_MESSAGE and PUT_MESSAGEX macro definitions to include file COPYMSG.REQ.
1 !        Move routine COPY$MSG_NUMBER to new module, COPYMSG.B32.
1 !        In CALC_ALQ, if /TRUNCATE was specified without /ALLOCATION, calculate allocation
1 !                value based on actual EOF of input file.
1 !        Add a global variable COPY$B_INCOMPAT. If this variable is set, don't output
1 !                incompatible attributes message because it has already been output once
1 !                for this output file.
1 !        In RMS_SETUP, when setting the MBC and MBF fields for a record mode copy,
1 !                set the MBC field to the size of the input file only the size is less than or
1 !                equal to 127 blocks. Otherwise, MBC goes negative.
1 !        In RMS_SETUP, a record mode copy from disk or tape loads RAB$W_USZ from XAB$W_LRL if
1 !                non-zero; otherwise, FAB$W_BLS.
1 !
1 !--
```

```
!++
! DETAILED FUNCTIONAL DESCRIPTION:
!
!       This utility program creates a copy of one or more user-specified
!       files.  These files can be explicitly named or can be referred to
!       through use of RMS wildcard file naming.  Two or more files may
!       optionally be concatenated to create a single output file.
!
!       All file I/O is done using standard RMS facilities.  Therefore,
!       the input and output files can exist on any device supported by RMS,
!       including devices at remote network nodes.  If possible, file copying
!       is done using block I/O.  Record I/O is used only when an input or
!       output file is record oriented (e.g., terminal, unit record) or when
!       a concatenated file is being copied.
!
!       This utility is intended to interface directly with a Command Language
!       Interpreter (CLI) and cannot be directly invoked from Command Language
!       level or from an executing program.  Numerous command options
!       (i.e., qualifiers) are supported to allow the Command Language user
!       to (1) optionally specify the location and attributes of the input
!       and output files, and (2) control the reporting of each file copy.
!
!       If more than one copy operation is specified in a single COPY request,
!       each file copy is performed independent of the others.  Therefore,
!       the failure of one file copy operation (e.g., I/O error, input
!       file not found) does not affect the remaining copy requests.  The
!       single exception to this rule is that unprocessed concatenated input
!       files are bypassed in the event of a file copy failure.
!
!--

!       NOTE:  This module contains some temporary code that (1) circumvents
!       a system problem or (2) cannot be implemented until an expected
!       system function is available.  In some cases, codes have been added;
!       in other cases, code has been "commented out".  In either case, each
!       statement affected includes a comment of the form "!#n", where "n" is
!       a number from the following table:
!
!               #1 - symbol not currently defined in STARLET.L32
!               #2 - I/O buffers cannot be locked in working set - known restriction
!               #3 - MODIFY does not accept FHC XAB - future feature
```

```
!
! TABLE OF CONTENTS:
!

FORWARD ROUTINE
    COPY$COPY,                                          ! Main COPY control routine
    COPY$CHECK_FILE_FOR_MATCH,                          ! Sees if input file matches command line criteria
    CREATE_DIR,                                         ! Create directory file
    RMS_SETUP,                                          ! RAB/buffer initialization
    COPY_FILE,                                          ! Copies an input file to the output file
    CLOSE_INFILE :  NOVALUE,                            ! Closes the current input file
    COPY$CLOSE_OUTF :  NOVALUE,                         ! Closes the current output file
    BYPASS_CONCAT,                                      ! Bypass concatenated input files after an error
    COPY$FIND_INPUT_FILE,                               ! Parse an input file-specification
    COPY$CALC_ALQ,                                      ! Calculate the output file allocation quantity
    REPORT_NAMES :  NOVALUE,                            ! Report names of input and output files
    REPORT_BYPASS :  NOVALUE,                           ! Report name of file bypassed
    COPY$LOG_MSG :  NOVALUE,                            ! Informational message routine
    COPY$INOPN_ERR :  NOVALUE,                          ! Input open error routine
    IN_READ_ERROR :  NOVALUE,                           ! Input read error routine
    IN_CLOSE_ERROR :  NOVALUE,                          ! Input close error routine
    COPY$OUTOPN_ERR :  NOVALUE,                         ! Output open error routine
    OUT_WRITE_ERROR :  NOVALUE,                         ! Output write error routine
    COPY$OCLOSE_ERR :  NOVALUE,                         ! Output close error routine
    COPY$MSG_NUMBER;                                    ! Compute message number

!
! INCLUDE FILES:
!

LIBRARY 'SYS$LIBRARY:STARLET.L32';                      ! VAX/VMS common definitions
REQUIRE 'SRC$:COPYMSG.REQ';                             ! Definition of macros to SIGNAL a message

!
! MACROS:
!

MACRO
    IN_NEQ_OUT[] =                                      ! Compare input and output FHC XAB field
        .INFILE_XABFHC[%REMAINING] NEQ .OUTFILE_XABFHC[%REMAINING] %,

    NAM$B_DVILNG = $DEFINE_BYTE[NAM$T_DVI] %,

    $DEFINE_BYTE( D, B, S, X ) = D, B, 8, 0 %,

    !
    ! Check to see if the global or local qualifier flag is set without the
    ! local negation flag being set.
    !
    qualifier_active( global_qual, local_qual, locally_negated ) =
        (IF (.global_qual AND NOT .locally_negated) OR .local_qual
         THEN true
         ELSE false )%
    ;


!
```

```
! EQUATED SYMBOLS:
!

LITERAL
    CLI_STATUS_LEN = 28,                                        ! Length of COPY$CLI_STATUS block
    SEM_STATUS_LEN =  4                                         ! Length of COPY$SEM_STATUS block
    ;
!    RME$K_OVERLAY = 0;                                         !#1 ***** KLUDGE *****

!
! Global variables
!

GLOBAL
    OUTFILE_COUNT :  INITIAL (0),                               ! Number of output files created

    BLOCK_COUNT,                                                ! Number of input blocks copied (current file)

    RECORD_COUNT,                                               ! Number of input records copied (current file)

    MOST_SEVERE_ERR :  BLOCK[4,BYTE]                            ! Most severe error encountered
                          INITIAL( SS$_NORMAL ),                !

    IO_BUFFER_BASE : INITIAL(0),                                ! Address of I/O buffer pool

    RMS_MBC : INITIAL(32),                                      ! Size of the RMS buffers

    BLOCK_SIZE,                                                 ! Input file block size

    COPY$CLI_STATUS : $BBLOCK[ CLI_STATUS_LEN ]                 ! Results of the command line parse
                        INITIAL(0),
    COPY$SEM_STATUS : $BBLOCK[ SEM_STATUS_LEN ]                 ! Status of the input and output files
                        INITIAL(0),
    COPY$B_INCOMPAT : BYTE INITIAL(0)                           ! Flag which is set if files have incoompatible attr
    ;

!
! YET ANOTHER REQUIRE FILE
!
REQUIRE
    'SRC$:COPY.REQ';                                            ! Field definitions for COPY$CLI_STATUS and COPY$SEM
```

```
; %PRINT:        File: VMSMAC.B32, Version V04-000, Edit 1, WWC, 09-JAN-1978
```

```
1 !
1 ! EXTERNAL REFERENCES:
1 !
1 EXTERNAL
1
1      !
1      ! Command line qualifier values
1      !
1      common_qual_context,                                        ! Common qualifier data area
1      curr_allocation_value,                                      ! The allocation for the output file
1      curr_protection_or,                                         ! Protection mask for /PROTECTION qualifier
1      curr_protection_and,                                        ! Protection mask for /PROTECTION qualifier
1
1      !
1      ! RMS definitions
1      !
1      infile_fab          : BLOCK [, BYTE],                       ! Input file FAB block
1      infile_rab          : BLOCK [, BYTE],                       ! Input file RAB block
1      infile_name         : VECTOR [, BYTE],                      ! Input file name after $OPEN
1      infile_xname        : VECTOR [, BYTE],                      ! Input file name before $OPEN
1      infile_nam_blk      : BLOCK [, BYTE],                       ! Primary input NAM block
1      infile_xabfhc       : BLOCK [, BYTE],                       ! File header characteristics XAB block
1      infile_xaball       : BLOCK [, BYTE],                       ! File allocation XAB block
1      infile_cli_desc     : $BBLOCK,                              ! Input file name on command line
1      in_name_desc        : VECTOR,                               ! Input file name descriptor
1      outfile_fab         : BLOCK [, BYTE],                       ! Output file FAB block
1      outfile_rab         : BLOCK [, BYTE],                       ! Output file RAB block
1      outfile_name        : VECTOR [, BYTE],                      ! Output file name after $OPEN
1      outfile_xname       : VECTOR [, BYTE],                      ! Output file name before $OPEN
1      outfile_nam_blk     : BLOCK [, BYTE],                       ! Output file NAM block
1      outfile_xabrdt      : BLOCK [, BYTE],                       ! Output file revision date/time XAB block
1      outfile_xabpro      : BLOCK [, BYTE],                       ! Output file protection XAB block
1      outfile_xabdat      : BLOCK [, BYTE],                       ! Output file date XAB block
1      outfile_xaball      : BLOCK [, BYTE],                       ! Output file allocation XAB block
1      outfile_xabfhc      : BLOCK [, BYTE],                       ! Output file file header characteristics XAB block
1      out_name_desc       : VECTOR;                               ! Output file name descriptor
1
1 EXTERNAL LITERAL
1      LIB$_FILFAIMAT,                                             ! File failed to match command line criteria
1      LIB$_QUIPRO                                                 ! User requested that processing cease
1      ;
1
1 EXTERNAL ROUTINE
1      COPY$GET_INFILE,                                            ! Gets the name of the input file
1      COPY$GET_OUTFIL,                                            ! Gets the name of the output file
1      COPY$OPN_INFILE,                                            ! Opens the input file
1      COPY$OPN_OUTFIL,                                            ! Opens an output file
1      CLI$GET_VALUE : ADDRESSING_MODE(GENERAL),                   ! Get a value from the command line
1      LIB$FIND_FILE : ADDRESSING_MODE(GENERAL),                   ! Find a file which fits the given filespec
1      LIB$GET_VM :  ADDRESSING_MODE(GENERAL),                     ! Virtual memory allocation
1      LIB$QUAL_FILE_MATCH : ADDRESSING_MODE(GENERAL),             ! Match a given file to the command line criteria
1      LIB$CHECK_DIR :  ADDRESSING_MODE(GENERAL),                  ! Determine if file is a directory
1      LIB$CREATE_DIR :  ADDRESSING_MODE(GENERAL);                 ! Create a directory file
```

```
1 ROUTINE COPY$COPY =                                               ! Primary COPY control routine
1
1 !++
1 ! FUNCTIONAL DESCRIPTION:
1 !
1 !       This routine is the primary control routine for the COPY utility.
1 !       It determines the basic logical flow and calls support routines
1 !       which perform each logical function.
1 !
1 ! FORMAL PARAMETERS:
1 !
1 !       AP.rlu.va - Argument list passed from the Command Language Interpreter
1 !
1 ! IMPLICIT INPUTS:
1 !
1 !       None
1 !
1 ! IMPLICIT OUTPUTS:
1 !
1 !       None
1 !
1 ! COMPLETION CODES:
1 !
1 !       Most severe error encountered during processing or SS$_NORMAL
1 !
1 ! SIDE EFFECTS:
1 !
1 !       None
1 !
1 !--
1
2    BEGIN
2
2    BUILTIN
2       AP;                                                         ! Declare the name of the argument pointer.
2
2    BIND
2       ARGUMENT_LIST = AP :  REF BLOCK[,BYTE];                     ! Declare the form of the argument list.
2
2    LOCAL
2       ptr,                                                        ! Temporary variables for character searching
2       address,
2       size,
2       STATUS;                                                     ! General routine return code
2
2
2
2
2 !
2 ! Get the output file-specification and all qualifiers from the CLI.
2 !
2
2    IF NOT COPY$GET_OUTFIL (                                       ! Get the output file spec from the CLI.
2                            OUTFILE_FAB,                           !    Specify the output FAB block address,
2                            OUTFILE_NAM_BLK,                       !    the output NAM block address,
2                            OUTFILE_XABFHC)                        !    and the output XABFHC block address.
```

```
	THEN
	    RETURN .MOST_SEVERE_ERR;				!   On error, return to CLI.

!
! The remainder of this routine is executed for each input
! file-specification supplied by the user.  Get the first input file.
!

	IF NOT (status = CLI$GET_VALUE( $DESCRIPTOR('INFILE'), infile_cli_desc))
	THEN
	    RETURN .status;

	WHILE 1 DO						! Beginning of repeat loop
	    BEGIN

!
! Get the next input file-specification from the CLI.  This routine call is a
! NOP if a wildcard file-specification is currently being processed;
! that is, a wildcard specification is repeatedly used until no furthur
! match is found.
!

	    STATUS = COPY$GET_INFILE (				! Get an input file-specification.
					INFILE_FAB,		!    Specify the address of the input FAB block,
					INFILE_NAM_BLK,		!    the address of the input NAM block,
					INFILE_XABALL);		!    and the address of the input XABALL block.
	    IF .STATUS EQL NO_MORE_FILES			! If there are no more input file-specs,
	    THEN						!
		EXITLOOP;					! exit the input file-spec processing loop.

	    IF .STATUS EQL OK					! If everything is OK so far,
	    THEN						!
		BEGIN						! begin normal input file processing.

!
! Open the current input file.
!

		STATUS = COPY$OPN_INFILE (INFILE_FAB);		! Open the current input file.

!
! If the input file is a directory file, then create the directory file
! on the output side if the file does not already exist.  If the output
! directory already exists, then do nothing.
!

		IF .status EQL ok			! If input opened ok,
		    AND lib$check_dir (infile_fab)	! and file is a directory,
		    AND NOT .outfile_fab [$FAB_DEV(sdi)] ! and not magtape output,
		THEN
		    IF NOT .outfile_fab[$FAB_DEV(net)]
			AND NOT .outfile_fab [$FAB_DEV(rec)] ! and not record device,
		    THEN
			BEGIN
			IF (.outfile_nam_blk[nam$v_exp_name] AND
				(NOT .outfile_nam_blk[nam$v_wild_name])) OR
```

```
; 6                              (.outfile_nam_blk[nam$v_exp_type] AND
; 5                                  (NOT .outfile_nam_blk[nam$v_wild_type])) OR
; 6                              (.outfile_nam_blk[nam$v_exp_ver] AND
; 6                                  (NOT .outfile_nam_blk[nam$v_wild_ver]))
; 5                          THEN
; 6                              BEGIN
; 6                              report_bypass(msg$_illdircopy);
; 6                              close_infile();                     ! Close input file
; 6                              END
; 5                          ELSE
; 6                              BEGIN
; 6                              status = create_dir (infile_fab, outfile_fab);
; 6                              IF .status EQL ss$_created  ! If file actually created,
; 6                              THEN
; 7                                  BEGIN
; 7                                  report_names();         ! Report file copied
; 7                                  outfile_count = .outfile_count + 1;
; 6                                  END;
; 6                              IF NOT .status                      ! If successful,
; 6                              THEN
; 7                                  report_bypass(msg$_notcopied); ! Else report failure
; 6                              close_infile();                     ! Close input file
; 6                              END
; 5                          END
; 4                  ELSE
; 5                          BEGIN
; 5                          report_bypass(msg$_dirnotcre);  ! Else report failure
; 5                          close_infile();                 ! Close input file
; 5                          END
; 4              ELSE
; 5              BEGIN
; 5
; 5  !
; 5  ! Create (or simply open) the output file (if it is not already open due to
; 5  ! input file concatenation) and then copy the entire input file to the
; 5  ! output file.
; 5  !
; 5              IF .STATUS EQL OK                       ! If the input file was successfully opened,
; 5              THEN                                    !
; 6                  BEGIN                               !
; 7                  IF (STATUS = COPY$OPN_OUTFIL (      ! create or open the output file unless it is
; 7                                              OUTFILE_FAB,
; 7                                              OUTFILE_RAB,
; 7                                              INFILE_FAB,
; 7                                              OUTFILE_COUNT))
; 7                                                      ! already open due to input concatenation.
; 6                  THEN
; 7                      BEGIN
; 8                      IF (STATUS = RMS_SETUP())           ! Setup the input and output RABs and buffers.
; 7                      THEN
; 8                          BEGIN
; 9                          IF (STATUS = COPY_FILE())       ! Copy the entire input file to the output file.
; 8                          THEN
; 9                              BEGIN
; 9                              IF .outfile_fab [$FAB_DEV(rec)]
```

```
                        AND NOT .outfile_fab [$FAB_DEV(net)]
                    THEN
                        BEGIN
                        size = .out_name_desc[0];
                        address = .out_name_desc[1];
                        ptr = CH$FIND_CH(.size,.address,':');
                        IF .ptr NEQ 0           ! If there is anything past the device, remove it
                        THEN
                            out_name_desc[0] =  .ptr - .address + 1;
                        END;
                    REPORT_NAMES()              ! Report the results if the copy was successful.
                    END
                ELSE                            ! Otherwise, report a partial copy.
                    REPORT_BYPASS( MSG$_NOTCMPLT );
                END
            ELSE
                REPORT_BYPASS( MSG$_NOTCOPIED );
            END
        ELSE                                    ! If the output file couldn't be opened,
            BEGIN
            !
            ! If this is an APPEND operation, then stop processing.
            ! There is no need to continue appending to a non-existant
            ! file.
            !
            IF .append_command
            THEN EXITLOOP;

            SELECTONE .status OF
            SET
                [ LIB$_FILFAIMAT ] :    ! Quietly skip this file
                        status = ok;
                [ LIB$_QUIPRO ]    :    ! User wishes to stop at this point
                        EXITLOOP;
                [ OTHERWISE ]      :    ! indicate the input file wasn't copied.
                        REPORT_BYPASS( MSG$_NOTCOPIED);
            TES;
            END; ! else stmt
        END;

    CLOSE_INFILE();                             ! Close the input file.

    END;                                        ! End of ELSE clause
    END;                                        ! End of processing a single input file specificatio

! If the user wishes to quit processing, then exit with a successful
! status.
!
IF .status EQL LIB$_QUIPRO
THEN
    status = ok;
!
! Bypass any concatenated input files if an error occurred during the
! file copy.
!
```

```
3          IF NOT .STATUS                                   ! If the input file was not successfully copied,
3          THEN                                             !
3              IF BYPASS_CONCAT()                           ! bypass any concatenated input files.
3              THEN
3                  EXITLOOP;
3      !
3      ! Close the output file unless another input file is to be
3      ! concatenated to the output file just written.
3      !
3
3          IF .MULTIPLE_OUTPUT AND NOT .APPEND_COMMAND      ! If multiple output files are being created,
3          THEN                                             !     and the command was not APPEND,
4              BEGIN
4              !
4              ! Set up protection if user specified explicitly.
4              !
4
5              IF qualifier_active( protect_qual, loc_protect_qual, neg_protect_qual )
4              THEN
5                  BEGIN
5                  outfile_xabpro [xab$w_pro] = .outfile_xabpro[ xab$w_pro] AND
5                                               .curr_protection_and;
5                  outfile_xabpro [xab$w_pro] = .outfile_xabpro[ xab$w_pro] OR
5                                               .curr_protection_or;
5                  END
4              ELSE
4                  outfile_xabrdt [xab$l_nxt] = 0;
4
4              !
4              ! Close the current output file.
4              !
4
4              COPY$CLOSE_OUTF();                           ! close the current output file, if any.
4
4              !
4              ! Reinitialize the XAB chain, since it may have been mucked with
4              ! by COPY$OPN_OUTFIL among other routines.
4              !
4
4              outfile_xaball [xab$l_nxt] = outfile_xabdat;
4              outfile_xabdat [xab$l_nxt] = outfile_xabrdt;
4              outfile_xabrdt [xab$l_nxt] = outfile_xabpro;
3              END;
3
2          END;                                             ! End of "WHILE 1 DO" input file-spec processing loo
2
2      !
2      ! Perform any necessary cleanup before exiting.
2      !
2
2      !
2      ! Set up protection if user specified explicitly.
2      !
2
3      IF qualifier_active( protect_qual, loc_protect_qual, neg_protect_qual )
2      THEN
3          BEGIN
```

```
3          outfile_xabpro [xab$w_pro] = .outfile_xabpro[ xab$w_pro] AND
3                                       .curr_protection_and;
3          outfile_xabpro [xab$w_pro] = .outfile_xabpro[ xab$w_pro] OR
3                                       .curr_protection_or;
3          END
2      ELSE
2          outfile_xabrdt [xab$l_nxt] = 0;
2
2      COPY$CLOSE_OUTF();                                       ! close the current output file, if any.
2
2      COPY$LOG_MSG( MSG$_NEWFILES );                           ! Report the number of files created.
2
2  !
2  ! Return to the caller.
2  !
2
2      RETURN .MOST_SEVERE_ERR;                                 ! Use the most severe error encountered
2                                                               ! as the completion code from this routine.
2
1      END;


                                                  .TITLE  COPYMAIN
                                                  .IDENT  \V04-000\

                                                  .PSECT  $PLIT$,NOWRT,NOEXE,2

        45  4C  49  46  4E  49  00000 P.AAB:      .ASCII  \INFILE\                               ;
                                00006             .BLKB   2
                     00000006   00008 P.AAA:      .LONG   6                                      ;
                     00000000'  0000C             .ADDRESS P.AAB                                 ;

                                                  .PSECT  $GLOBAL$,NOEXE,2

                     00000000   00000 OUTFILE_COUNT::
                                                  .LONG   0                                      ;
                                00004 BLOCK_COUNT::
                                                  .BLKB   4
                                00008 RECORD_COUNT::
                                                  .BLKB   4
                     00000001   0000C MOST_SEVERE_ERR::
                                                  .LONG   1                                      ;
                     00000000   00010 IO_BUFFER_BASE::
                                                  .LONG   0                                      ;
                     00000020   00014 RMS_MBC::
                                                  .LONG   32                                     ;
                                00018 BLOCK_SIZE::
                                                  .BLKB   4
                     00000000   0001C COPY$CLI_STATUS::
                                                  .LONG   0                                      ;
                                00020             .BLKB   24
                     00000000   00038 COPY$SEM_STATUS::
                                                  .LONG   0                                      ;
                           00   0003C COPY$B_INCOMPAT::
                                                  .BYTE   0                                      ;

                                                  .EXTRN  CLI$_PRESENT, CLI$_NEGATED
```

```
                                                      .EXTRN   CLI$_LOCPRES, CLI$_LOCNEG
                                                      .EXTRN   COMMON_QUAL_CONTEXT
                                                      .EXTRN   CURR_ALLOCATION_VALUE
                                                      .EXTRN   CURR_PROTECTION_OR
                                                      .EXTRN   CURR_PROTECTION_AND
                                                      .EXTRN   INFILE_FAB, INFILE_RAB
                                                      .EXTRN   INFILE_NAME, INFILE_XNAME
                                                      .EXTRN   INFILE_NAM_BLK, INFILE_XABFHC
                                                      .EXTRN   INFILE_XABALL, INFILE_CLI_DESC
                                                      .EXTRN   IN_NAME_DESC, OUTFILE_FAB
                                                      .EXTRN   OUTFILE_RAB, OUTFILE_NAME
                                                      .EXTRN   OUTFILE_XNAME, OUTFILE_NAM_BLK
                                                      .EXTRN   OUTFILE_XABRDT, OUTFILE_XABPRO
                                                      .EXTRN   OUTFILE_XABDAT, OUTFILE_XABALL
                                                      .EXTRN   OUTFILE_XABFHC, OUT_NAME_DESC
                                                      .EXTRN   LIB$_FILFAIMAT, LIB$_QUIPRO
                                                      .EXTRN   COPY$GET_INFILE
                                                      .EXTRN   COPY$GET_OUTFIL
                                                      .EXTRN   COPY$OPN_INFILE
                                                      .EXTRN   COPY$OPN_OUTFIL
                                                      .EXTRN   CLI$GET_VALUE, LIB$FIND_FILE
                                                      .EXTRN   LIB$GET_VM, LIB$QUAL_FILE_MATCH
                                                      .EXTRN   LIB$CHECK_DIR, LIB$CREATE_DIR

                                                      .PSECT   $CODE$,NOWRT,2

                                 0FFC 00000 COPY$COPY:
                                                      .WORD    Save R2,R3,R4,R5,R6,R7,R8,R9,R10,R11       ; 0995
             5B      0000G   CF   9E 00002            MOVAB    OUTFILE_XABPRO+8, R11                      ;
             5A      0000G   CF   9E 00007            MOVAB    INFILE_FAB, R10                            ;
             59      0000G   CF   9E 0000C            MOVAB    OUTFILE_NAM_BLK+52, R9                     ;
             58      0000G   CF   9E 00011            MOVAB    OUTFILE_FAB+64, R8                         ;
             57      0000'   CF   9E 00016            MOVAB    COPY$CLI_STATUS+2, R7                      ;
                     0000G   CF   9F 0001B            PUSHAB   OUTFILE_XABFHC                             ; 1048
                        CC   A9   9F 0001F            PUSHAB   OUTFILE_NAM_BLK                            ;
                        C0   A8   9F 00022            PUSHAB   OUTFILE_FAB                                ;
     0000G   CF                03 FB 00025            CALLS    #3, COPY$GET_OUTFIL                        ;
             03                50 E8 0002A            BLBS     R0, 1$                                     ;
                             01D1 31 0002D            BRW      38$                                        ;
                     0000G   CF   9F 00030 1$:        PUSHAB   INFILE_CLI_DESC                            ; 1060
                     0000'   CF   9F 00034            PUSHAB   P.AAA                                      ;
 00000000G   00                02 FB 00038            CALLS    #2, CLI$GET_VALUE                          ;
             52                50 D0 0003F            MOVL     R0, STATUS                                 ;
             04                52 E8 00042            BLBS     STATUS, 2$                                 ;
             50                52 D0 00045            MOVL     STATUS, R0                                 ; 1062
                                  04 00048            RET                                                 ;
                     0000G   CF   9F 00049 2$:        PUSHAB   INFILE_XABALL                              ; 1074
                     0000G   CF   9F 0004D            PUSHAB   INFILE_NAM_BLK                             ;
                               5A DD 00051            PUSHL    R10                                        ;
     0000G   CF                03 FB 00053            CALLS    #3, COPY$GET_INFILE                        ;
             52                50 D0 00058            MOVL     R0, STATUS                                 ;
             03                52 D1 0005B            CMPL     STATUS, #3                                 ; 1079
                               03 12 0005E            BNEQ     3$                                         ;
                             016D 31 00060            BRW      33$                                        ;
             01                52 D1 00063 3$:        CMPL     STATUS, #1                                 ; 1083
                               03 13 00066            BEQL     4$                                         ;
                             0108 31 00068            BRW      25$                                        ;
```

```
                                  5A  DD 0006B 4$:      PUSHL   R10                                   ; 1091
            0000G  CF             01  FB 0006D          CALLS   #1, COPY$OPN_INFILE                   ;
                   52             50  D0 00072          MOVL    R0, STATUS                            ;
                                  53  D4 00075          CLRL    R3                                    ; 1099
                   01             52  D1 00077          CMPL    STATUS, #1                            ;
                                  63  12 0007A          BNEQ    12$                                   ;
                                  53  D6 0007C          INCL    R3                                    ;
                                  5A  DD 0007E          PUSHL   R10                                   ; 1100
        00000000G  00             01  FB 00080          CALLS   #1, LIB$CHECK_DIR                     ;
                   55             50  E9 00087          BLBC    R0, 12$                               ;
  51               68             04  E0 0008A          BBS     #4, OUTFILE_FAB+64, 12$               ; 1101
  45        01     A8             05  E0 0008E          BBS     #5, OUTFILE_FAB+65, 10$               ; 1103
                   42             68  E8 00093          BLBS    OUTFILE_FAB+64, 10$                   ; 1104
  04               69             02  E1 00096          BBC     #2, OUTFILE_NAM_BLK+52, 5$            ; 1107
  0F               69             05  E1 0009A          BBC     #5, OUTFILE_NAM_BLK+52, 7$            ; 1108
  04               69             01  E1 0009E 5$:      BBC     #1, OUTFILE_NAM_BLK+52, 6$            ; 1109
  07               69             04  E1 000A2          BBC     #4, OUTFILE_NAM_BLK+52, 7$            ; 1110
                   0B             69  E9 000A6 6$:      BLBC    OUTFILE_NAM_BLK+52, 8$                ; 1111
  07               69             03  E0 000A9          BBS     #3, OUTFILE_NAM_BLK+52, 8$            ; 1112
                   7E      12E8   8F  3C 000AD 7$:      MOVZWL  #4840, -(SP)                          ; 1115
                                  29  11 000B2          BRB     11$                                   ;
                             C0   A8  9F 000B4 8$:      PUSHAB  OUTFILE_FAB                           ; 1120
                                  5A  DD 000B7          PUSHL   R10                                   ;
            0000V  CF             02  FB 000B9          CALLS   #2, CREATE_DIR                        ;
                   52             50  D0 000BE          MOVL    R0, STATUS                            ;
        00000619   8F             52  D1 000C1          CMPL    STATUS, #1561                         ; 1121
                                  08  12 000C8          BNEQ    9$                                    ;
            0000V  CF             00  FB 000CA          CALLS   #0, REPORT_NAMES                      ; 1124
                             E2   A7  D6 000CF          INCL    OUTFILE_COUNT                         ; 1125
                   68             52  E8 000D2 9$:      BLBS    STATUS, 15$                           ; 1127
                               008C   31 000D5          BRW     22$                                   ; 1129
                   7E      12C0   8F  3C 000D8 10$:     MOVZWL  #4800, -(SP)                          ; 1135
                                  65  11 000DD 11$:     BRB     17$                                   ;
                   77             53  E9 000DF 12$:     BLBC    R3, 20$                               ; 1148
                             E2   A7  9F 000E2          PUSHAB  OUTFILE_COUNT                         ; 1151
                                  5A  DD 000E5          PUSHL   R10                                   ;
                          0000G   CF  9F 000E7          PUSHAB  OUTFILE_RAB                           ;
                             C0   A8  9F 000EB          PUSHAB  OUTFILE_FAB                           ;
            0000G  CF             04  FB 000EE          CALLS   #4, COPY$OPN_OUTFIL                   ;
                   52             50  D0 000F3          MOVL    R0, STATUS                            ;
                   4D             52  E9 000F6          BLBC    STATUS, 18$                           ;
            0000V  CF             00  FB 000F9          CALLS   #0, RMS_SETUP                         ; 1159
                   52             50  D0 000FE          MOVL    R0, STATUS                            ;
                   60             52  E9 00101          BLBC    STATUS, 22$                           ;
            0000V  CF             00  FB 00104          CALLS   #0, COPY_FILE                         ; 1162
                   52             50  D0 00109          MOVL    R0, STATUS                            ;
                   30             52  E9 0010C          BLBC    STATUS, 16$                           ;
                   26             68  E9 0010F          BLBC    OUTFILE_FAB+64, 14$                   ; 1165
  21        01     A8             05  E0 00112          BBS     #5, OUTFILE_FAB+65, 14$               ; 1166
                   55     0000G   CF  D0 00117          MOVL    OUT_NAME_DESC, SIZE                   ; 1169
                   54     0000G   CF  D0 0011C          MOVL    OUT_NAME_DESC+4, ADDRESS              ; 1170
  64               55             3A  3A 00121          LOCC    #58, SIZE, (ADDRESS)                  ; 1171
                                  02  12 00125          BNEQ    13$                                   ;
                                  51  D4 00127          CLRL    R1                                    ;
                   56             51  D0 00129 13$:     MOVL    R1, PTR                               ;
                                  0A  13 0012C          BEQL    14$                                   ; 1172
  51               56             54  C3 0012E          SUBL3   ADDRESS, PTR, R1                      ; 1174
```

```
                                0000G  CF          01  A1 9E 00132         MOVAB   1(R1), OUT_NAME_DESC                    ;
                                0000V  CF              00 FB 00138 14$:    CALLS   #0, REPORT_NAMES                        ; 1176
                                                       2F 11 0013D 15$:    BRB     24$                                     ;
                                       7E        11C0  8F 3C 0013F 16$:    MOVZWL  #4544, -(SP)                            ; 1179
                                                       23 11 00144 17$:    BRB     23$                                     ;
                                       03          FE  A7 E9 00146 18$:    BLBC    COPY$CLI_STATUS, 19$                    ; 1191
                                                     0083 31 0014A         BRW     33$                                     ;
                            00000000G  8F              52 D1 0014D 19$:    CMPL    STATUS, #LIB$_FILFAIMAT                 ; 1196
                                                       05 12 00154         BNEQ    21$                                     ;
                                       52              01 D0 00156         MOVL    #1, STATUS                              ; 1197
                                                       13 11 00159 20$:    BRB     24$                                     ;
                            00000000G  8F              52 D1 0015B 21$:    CMPL    STATUS, #LIB$_QUIPRO                    ; 1198
                                                       6C 13 00162         BEQL    33$                                     ;
                                       7E        11B8  8F 3C 00164 22$:    MOVZWL  #4536, -(SP)                            ; 1201
                                0000V  CF              01 FB 00169 23$:    CALLS   #1, REPORT_BYPASS                       ;
                                0000V  CF              00 FB 0016E 24$:    CALLS   #0, CLOSE_INFILE                        ; 1206
                            00000000G  8F              52 D1 00173 25$:    CMPL    STATUS, #LIB$_QUIPRO                    ; 1215
                                                       03 12 0017A         BNEQ    26$                                     ;
                                       52              01 D0 0017C         MOVL    #1, STATUS                              ; 1217
                                       08              52 E8 0017F 26$:    BLBS    STATUS, 27$                             ; 1223
                                0000V  CF              00 FB 00182         CALLS   #0, BYPASS_CONCAT                       ; 1225
                                       46              50 E8 00187         BLBS    R0, 33$                                 ;
                                       3F          1B  A7 E9 0018A 27$:    BLBC    COPY$SEM_STATUS+1, 32$                  ; 1233
                                       3B          FE  A7 E8 0018E         BLBS    COPY$CLI_STATUS, 32$                    ;
                        05         01  A7              05 E1 00192         BBC     #5, COPY$CLI_STATUS+3, 28$              ; 1240
                                                   01  A7 95 00197         TSTB    COPY$CLI_STATUS+3                       ;
                                                       05 18 0019A         BGEQ    29$                                     ;
                        0F         01  A7              06 E1 0019C 28$:    BBC     #6, COPY$CLI_STATUS+3, 30$              ;
                                       50       0000G  CF D2 001A1 29$:    MCOML   CURR_PROTECTION_AND, R0                 ; 1244
                                       6B              50 AA 001A6         BICW2   R0, OUTFILE_XABPRO+8                    ;
                                       6B       0000G  CF A8 001A9         BISW2   CURR_PROTECTION_OR, OUTFILE_XABPRO+8    ; 1246
                                                       04 11 001AE         BRB     31$                                     ; 1240
                                                0000G  CF D4 001B0 30$:    CLRL    OUTFILE_XABRDT+4                        ; 1249
                                0000V  CF              00 FB 001B4 31$:    CALLS   #0, COPY$CLOSE_OUTF                     ; 1255
                                0000G  CF       0000G  CF 9E 001B9         MOVAB   OUTFILE_XABDAT, OUTFILE_XABALL+4        ; 1262
                                0000G  CF       0000G  CF 9E 001C0         MOVAB   OUTFILE_XABRDT, OUTFILE_XABDAT+4        ; 1263
                                0000G  CF          F8  AB 9E 001C7         MOVAB   OUTFILE_XABPRO, OUTFILE_XABRDT+4        ; 1264
                                                     FE79 31 001CD 32$:    BRW     2$                                      ; 1064
                        05         01  A7              05 E1 001D0 33$:    BBC     #5, COPY$CLI_STATUS+3, 34$              ; 1277
                                                   01  A7 95 001D5         TSTB    COPY$CLI_STATUS+3                       ;
                                                       05 18 001D8         BGEQ    35$                                     ;
                        0F         01  A7              06 E1 001DA 34$:    BBC     #6, COPY$CLI_STATUS+3, 36$              ;
                                       50       0000G  CF D2 001DF 35$:    MCOML   CURR_PROTECTION_AND, R0                 ; 1281
                                       6B              50 AA 001E4         BICW2   R0, OUTFILE_XABPRO+8                    ;
                                       6B       0000G  CF A8 001E7         BISW2   CURR_PROTECTION_OR, OUTFILE_XABPRO+8    ; 1283
                                                       04 11 001EC         BRB     37$                                     ; 1277
                                                0000G  CF D4 001EE 36$:    CLRL    OUTFILE_XABRDT+4                        ; 1286
                                0000V  CF              00 FB 001F2 37$:    CALLS   #0, COPY$CLOSE_OUTF                     ; 1288
                                       7E        1091  8F 3C 001F7         MOVZWL  #4241, -(SP)                            ; 1290
                                0000V  CF              01 FB 001FC         CALLS   #1, COPY$LOG_MSG                        ;
                                       50          EE  A7 D0 00201 38$:    MOVL    MOST_SEVERE_ERR, R0                     ; 1296
                                                          04 00205         RET                                             ; 1299

; Routine Size:  518 bytes,    Routine Base:  $CODE$ + 0000
;  1 GLOBAL ROUTINE COPY$CHECK_FILE_FOR_MATCH =
;  1
;  1 !++
;  1 !
;  1 ! FUNCTIONAL DESCRIPTION:
;  1 !
;  1 !       This routine sets up the parameters for and calls LIB$QUAL_FILE_MATCH to see if the input
;  1 !       file matches the criteria given on the command line.
;  1 !
;  1 ! FORMAL PARAMETERS:
;  1 !
;  1 !       None
;  1 !
;  1 ! IMPLICIT INPUTS:
;  1 !
;  1 !       IN_NAME_DESC    : Input file name descriptor
;  1 !       OUT_NAME_DESC   : Output file name descriptor
;  1 !       OUTFILE_OPEN    : Output file is currently open
;  1 !       COMMON_QUAL_CONTEXT : Common qualifier data area
;  1 !
;  1 ! IMPLICIT OUTPUTS:
;  1 !
;  1 !       None
;  1 !
;  1 ! ROUTINE VALUE:
;  1 !
;  1 !       Whatever LIB$QUAL_FILE_MATCH returns.
;  1 !
;  1 ! COMPLETION CODES:
;  1 !
;  1 !       None
;  1 !
;  1 ! SIDE EFFECTS:
;  1 !
;  1 !       None
;  1 !
;  1 !--
;  1
;  2 BEGIN
;  2
;  2 LOCAL
;  2     out_desc     :                        ! Temporary desc. for output file name
;  2                     VECTOR[ 2 ],
;  2     prompt_string_desc,                   ! Desc. for /CONFIRM prompt string address
;  2     prompt_args  :                        ! Argument list for /CONFIRM prompt
;  2                     VECTOR[ 2 ]
;  2     ;
;  2
;  2
;  2
;  2 ! Pick to appropriate propmt string, depending on whether the input file is
;  2 ! being append to an output file or not.
;  2 !
;  2 IF .append_command OR  .outfile_open
;  2     THEN prompt_string_desc = $DESCRIPTOR('Append !AS to !AS? [N]')
;  2     ELSE prompt_string_desc = $DESCRIPTOR('Copy !AS to !AS? [N]');
;  2
;   828          1357  2
;   829          1358  2  ! File in the file name descriptors.
;   830          1359  2  !
;   831          1360  2  prompt_args[ 0 ] = in_name_desc;
;   832          1361  2  prompt_args[ 1 ] = out_desc;
;   833          1362  2
;   834          1363  2  IF .outfile_nam_blk[ NAM$B_RSL ] NEQ 0
;   835          1364  2  THEN
;   836          1365  3      BEGIN
;   837          1366  3      out_desc[ 0 ] = .outfile_nam_blk[ NAM$B_RSL ];
;   838          1367  3      out_desc[ 1 ] = outfile_name;
;   839          1368  3      END
;   840          1369  2  ELSE
;   841          1370  2      IF .outfile_nam_blk[ NAM$B_ESL ] NEQ 0
;   842          1371  2      THEN
;   843          1372  3          BEGIN
;   844          1373  3          out_desc[ 0 ] = .outfile_nam_blk[ NAM$B_ESL ];
;   845          1374  3          out_desc[ 1 ] = outfile_xname;
;   846          1375  3          END
;   847          1376  2      ELSE
;   848          1377  2          prompt_args[ 1 ] = out_name_desc;
;   849          1378  2
;   850          1379  2  !
;   851          1380  2  ! Compare the current input file to the command line criteria.  Return the
;   852          1381  2  ! results of the comparison to the calling routine.
;   853          1382  2  !
;   854          1383  2  RETURN LIB$QUAL_FILE_MATCH( common_qual_context, infile_fab, 0,
;   855          1384  2      .prompt_string_desc, prompt_args, 0);
;   856          1385  2
;   857          1386  1  END;                                    ! End of routine COPY$CHECK_FILE_FOR_MATCH


                                                                            .PSECT  $PLIT$,NOWRT,NOEXE,2

21  20  6F  74  20  53  41  21  20  64  6E  65  70  70  41  00010 P.AAD:   .ASCII  \Append !AS to !AS? [N]\          ;
                                5D  4E  5B  20  3F  53  41  0001F                                                     ;
                                                            00026          .BLKB   2
                                                00000016    00028 P.AAC:   .LONG   22                                ;
                                                00000000'   0002C          .ADDRESS P.AAD                            ;
53  41  21  20  6F  74  20  53  41  21  20  79  70  6F  43  00030 P.AAF:   .ASCII  \Copy !AS to !AS? [N]\            ;
                                5D  4E  5B  20  3F          0003F                                                     ;
                                                00000014    00044 P.AAE:   .LONG   20                                ;
                                                00000000'   00048          .ADDRESS P.AAF                            ;


                                                                            .PSECT  $CODE$,NOWRT,2

                                                    0000  00000            .ENTRY  COPY$CHECK_FILE_FOR_MATCH, Save nothing     ; 1300
                              5E              10     C2  00002             SUBL2   #16, SP                                     ;
                              06     0000'    CF     E8  00005             BLBS    COPY$CLI_STATUS, 1$                         ; 1353
                 07   0000'   CF              01     E1  0000A             BBC     #1, COPY$SEM_STATUS+2, 2$                   ;
                              51     0000'    CF     9E  00010 1$:         MOVAB   P.AAC, PROMPT_STRING_DESC                   ; 1354
                                              05     11  00015             BRB     3$                                          ;
                              51     0000'    CF     9E  00017 2$:         MOVAB   P.AAE, PROMPT_STRING_DESC                   ; 1355
                              6E     0000G    CF     9E  0001C 3$:         MOVAB   IN_NAME_DESC, PROMPT_ARGS                   ; 1360
```

```
                        04    AE      08   AE   9E 00021          MOVAB   OUT_DESC, PROMPT_ARGS+4                  ; 1361
                              50   0000G   CF   9A 00026          MOVZBL  OUTFILE_NAM_BLK+3, R0                    ; 1363
                                           0C   13 0002B          BEQL    4$                                       ;
                        03    AE           50   D0 0002D          MOVL    R0, OUT_DESC                             ; 1366
                        0C    AE   0000G   CF   9E 00031          MOVAB   OUTFILE_NAME, OUT_DESC+4                 ; 1367
                                           19   11 00037          BRB     6$                                       ; 1363
                              50   0000G   CF   9A 00039 4$:      MOVZBL  OUTFILE_NAM_BLK+11, R0                   ; 1370
                                           0C   13 0003E          BEQL    5$                                       ;
                        08    AE           50   D0 00040          MOVL    R0, OUT_DESC                             ; 1373
                        0C    AE   0000G   CF   9E 00044          MOVAB   OUTFILE_XNAME, OUT_DESC+4                ; 1374
                                           06   11 0004A          BRB     6$                                       ; 1370
                        04    AE   0000G   CF   9E 0004C 5$:      MOVAB   OUT_NAME_DESC, PROMPT_ARGS+4             ; 1377
                                           7E   D4 00052 6$:      CLRL    -(SP)                                    ; 1383
                                      04   AE   9F 00054          PUSHAB  PROMPT_ARGS                              ;
                                           51   DD 00057          PUSHL   PROMPT_STRING_DESC                       ; 1384
                                           7E   D4 00059          CLRL    -(SP)                                    ; 1383
                                   0000G   CF   9F 0005B          PUSHAB  INFILE_FAB                               ;
                                   0000G   CF   9F 0005F          PUSHAB  COMMON_QUAL_CONTEXT                      ;
                 00000000G    00           06   FB 00063          CALLS   #6, LIB$QUAL_FILE_MATCH                  ;
                                                04 0006A          RET                                              ; 1386

; Routine Size:  107 bytes,    Routine Base:  $CODE$ + 0206
```

```
1 ROUTINE CREATE_DIR (input_fab, output_fab) =
1
1 !---
1 !
1 !       This routine is called to create a directory file on
1 !       the output side if the directory does not already exist.
1 !       If the directory already exists, do nothing.
1 !
1 ! Inputs:
1 !
1 !       input_fab = Address of FAB describing opened directory file
1 !       output_fab = Address of FAB describing the device and directory
1 !                       into which the directory file should be created.
1 !
1 ! Outputs:
1 !
1 !       Routine value = status return
1 !---
1
2 BEGIN
2
2 MAP
2     input_fab:  REF BLOCK[,BYTE],        ! Input FAB
2     output_fab: REF BLOCK[,BYTE];        ! Output FAB
2
2 BIND
2     input_nam = .input_fab [fab$l_nam]: BLOCK[,BYTE],
2     output_nam = .output_fab [fab$l_nam]: BLOCK[,BYTE];
2
2 LOCAL
2     ptr,                                  ! String temporary pointer
2     addr,size,                            ! descriptor of search string
2     buffer:      VECTOR [nam$c_maxrss,BYTE], ! file spec buffer
2     bufdesc:     VECTOR [2],              ! descriptor of above buffer
2     terminator: BYTE,                     ! Directory spec. terminator
2     status;                               ! status variable
2
2 record_count = 0;                         ! Initialize the record count
2 block_count  = 0;                         ! Initialize the block count
2
2 status = $RMS_PARSE (FAB = .output_fab); ! Get full name of directory file
2
2 size = .output_nam [nam$b_esl];           ! Get output expanded name
2 addr = .output_nam [nam$l_esa];
2
2 IF NOT .status
2 THEN
3     BEGIN
3     put_messagex(.status);
3     RETURN .status;
2     END;
2
2 ptr = CH$FIND_CH(.size, .addr, ']');     ! Find end of directory spec
2 IF .ptr EQL 0                             ! If not found,
2 THEN
3     BEGIN
3     ptr = CH$FIND_CH(.size, .addr, '>');   ! Alternate syntax
```

```
;  3         IF .ptr EQL 0                         ! If still not found,
;  3         THEN
;  3             put_message(rms$_esa);            ! return invalid expanded string
;  3         END;
;  2
;  2     size = .ptr + 1 - .addr;                  ! Figure length of device and dir.
;  2     CH$MOVE(.size, .addr, buffer);            ! Copy device and directory into buffer
;  2     terminator = .buffer [.size-1];           ! Remember terminator on dir. spec.
;  2     buffer [.size-1] = '.';                   ! and overwrite it with '.'
;  2
;  2     bufdesc [0] = .size;                      ! Setup buffer descriptor
;  2     bufdesc [1] = buffer;
;  2
;  2     size = .input_nam [nam$b_rsl];            ! Get input result name
;  2     addr = .input_nam [nam$l_rsa];
;  2
;  2     ptr = CH$FIND_CH(.size, .addr, ']');      ! Find start of file name on input side
;  2     IF .ptr EQL 0                             ! If not found,
;  2     THEN
;  3         BEGIN
;  3         ptr = CH$FIND_CH(.size, .addr, '>'); ! Alternate syntax
;  3         IF .ptr EQL 0                         ! If still not found
;  3         THEN
;  3             put_message(rms$_esa);            ! return invalid expanded string
;  2         END;
;  2
;  2     size = .size - (.ptr + 1 - .addr);        ! Figure descriptor of file name
;  2     addr = .ptr + 1;
;  2
;  2     ptr = CH$FIND_CH(.size, .addr, '.');      ! Find where file name ends
;  2     IF .ptr EQL 0                             ! If not found,
;  2     THEN
;  2         RETURN rms$_esa;                      ! return invalid expanded string
;  2     size = .ptr - .addr;                      ! Figure descriptor of file name only
;  2
;  2     CH$MOVE(.size, .addr, buffer+.bufdesc[0]); ! Append subdirectory name to buffer
;  2     buffer [.bufdesc[0]+.size] = .terminator;  ! Tack terminator on end of it
;  2     bufdesc [0] = .bufdesc[0] + .size + 1;  ! Update string descriptor
;  2
;  2     out_name_desc [0] = .bufdesc [0];         ! Copy length of string
;  2     CH$MOVE(.bufdesc[0], .bufdesc[1], .out_name_desc[1]);    ! and string too
;  2
;  2     status = LIB$CREATE_DIR (bufdesc);        ! Create directory file with defaults
;  2
;  2     IF NOT .status                            ! If error detected,
;  2     THEN
;  2         put_messagex(.status);                ! then signal status
;  2
;  2     RETURN .status;                           ! return with status
;  2
;  1     END;


                                                  .EXTRN  SYS$PARSE

                                       OFFC 00000 CREATE_DIR:
```

```
                                                .WORD    Save R2,R3,R4,R5,R6,R7,R8,R9,R10,R11  ; 1387
                   5E     FEF8 CE 9E 00002      MOVAB    -264(SP), SP                          ;
                   50       04 AC D0 00007      MOVL     INPUT_FAB, R0                         ; 1413
                   58       28 A0 D0 0000B      MOVL     40(R0), R8                            ;
                   50       08 AC D0 0000F      MOVL     OUTPUT_FAB, R0                        ; 1414
                   52       28 A0 D0 00013      MOVL     40(R0), R2                            ;
                         0000' CF 7C 00017      CLRQ     BLOCK_COUNT                           ; 1425
                               50 DD 0001B      PUSHL    R0                                    ; 1427
         00000000G 00          01 FB 0001D      CALLS    #1, SYS$PARSE                         ;
                   5A          50 D0 00024      MOVL     R0, STATUS                            ;
                   57       0B A2 9A 00027      MOVZBL   11(R2), SIZE                          ; 1429
                   56       0C A2 D0 0002B      MOVL     12(R2), ADDR                          ; 1430
                   03          5A E8 0002F      BLBS     STATUS, 1$                            ; 1432
                          00DD 31 00032      BRW      10$                                   ;
     66            57       5D 8F 3A 00035 1$:  LOCC     #93, SIZE, (ADDR)                     ; 1439
                               02 12 0003A      BNEQ     2$                                    ;
                               51 D4 0003C      CLRL     R1                                    ;
                   59          51 D0 0003E 2$:  MOVL     R1, PTR                               ;
                               21 12 00041      BNEQ     4$                                    ; 1440
     66            57          3E 3A 00043      LOCC     #62, SIZE, (ADDR)                     ; 1443
                               02 12 00047      BNEQ     3$                                    ;
                               51 D4 00049      CLRL     R1                                    ;
                   59          51 D0 0004B 3$:  MOVL     R1, PTR                               ;
                               14 12 0004E      BNEQ     4$                                    ; 1444
                      000184FC 8F DD 00050      PUSHL    #99580                                ; 1446
             0000V CF          01 FB 00056      CALLS    #1, COPY$MSG_NUMBER                   ;
                               50 DD 0005B      PUSHL    R0                                    ;
         00000000G 00          01 FB 0005D      CALLS    #1, LIB$STOP                          ;
     51            59          56 C3 00064 4$:  SUBL3    ADDR, PTR, R1                         ; 1449
                   57       01 A1 9E 00068      MOVAB    1(R1), SIZE                           ;
08   AE            66          57 28 0006C      MOVC3    SIZE, (ADDR), BUFFER                  ; 1450
                   58     07 AE47 90 00071      MOVB     BUFFER-1[SIZE], TERMINATOR            ; 1451
              07 AE47          2E 90 00076      MOVB     #46, BUFFER-1[SIZE]                   ; 1452
                   6E          57 D0 0007B      MOVL     SIZE, BUFDESC                         ; 1454
                04 AE       08 AE 9E 0007E      MOVAB    BUFFER, BUFDESC+4                     ; 1455
                   57       03 A8 9A 00083      MOVZBL   3(R8), SIZE                           ; 1457
                   56       04 A8 D0 00087      MOVL     4(R8), ADDR                           ; 1458
     66            57       5D 8F 3A 0008B      LOCC     #93, SIZE, (ADDR)                     ; 1460
                               02 12 00090      BNEQ     5$                                    ;
                               51 D4 00092      CLRL     R1                                    ;
                   59          51 D0 00094 5$:  MOVL     R1, PTR                               ;
                               21 12 00097      BNEQ     7$                                    ; 1461
     66            57          3E 3A 00099      LOCC     #62, SIZE, (ADDR)                     ; 1464
                               02 12 0009D      BNEQ     6$                                    ;
                               51 D4 0009F      CLRL     R1                                    ;
                   59          51 D0 000A1 6$:  MOVL     R1, PTR                               ;
                               14 12 000A4      BNEQ     7$                                    ; 1465
                      000184FC 8F DD 000A6      PUSHL    #99580                                ; 1467
             0000V CF          01 FB 000AC      CALLS    #1, COPY$MSG_NUMBER                   ;
                               50 DD 000B1      PUSHL    R0                                    ;
         00000000G 00          01 FB 000B3      CALLS    #1, LIB$STOP                          ;
     50            56          59 C3 000BA 7$:  SUBL3    PTR, ADDR, R0                         ; 1470
                   57     FF A047 9E 000BE      MOVAB    -1(R0)[SIZE], SIZE                    ;
                   56       01 A9 9E 000C3      MOVAB    1(R9), ADDR                           ; 1471
     66            57          2E 3A 000C7      LOCC     #46, SIZE, (ADDR)                     ; 1473
                               02 12 000CB      BNEQ     8$                                    ;
                               51 D4 000CD      CLRL     R1                                    ;
```

```
                                      59          51  D0 000CF 8$:      MOVL    R1, PTR                                          ;
                                                  08  12 000D2          BNEQ    9$                                               ; 1474
                                   50 000184FC    8F  D0 000D4          MOVL    #99580, R0                                       ; 1476
                                                      04 000DB          RET                                                      ;
                        57            59          56  C3 000DC 9$:      SUBL3   ADDR, PTR, SIZE                                  ; 1477
                                      50    08    AE  9E 000E0          MOVAB   BUFFER, R0                                       ; 1479
                  00 BE40             66          57  28 000E4          MOVC3   SIZE, (ADDR), @BUFDESC[R0]                       ;
                        50            6E          57  C1 000EA          ADDL3   SIZE, BUFDESC, R0                                ; 1480
                              08 AE40             5B  90 000EE          MOVB    TERMINATOR, BUFFER[R0]                           ;
                                      6E    01    A0  9E 000F3          MOVAB   1(R0), BUFDESC                                   ; 1481
                               0000G  CF          6E  D0 000F7          MOVL    BUFDESC, OUT_NAME_DESC                           ; 1483
                 0000G  DF       04   BE          6E  28 000FC          MOVC3   BUFDESC, @BUFDESC+4, @OUT_NAME_DESC+4            ; 1484
                                                  5E  DD 00103          PUSHL   SP                                               ; 1486
                           00000000G  00          01  FB 00105          CALLS   #1, LIB$CREATE_DIR                               ;
                                      5A          50  D0 0010C          MOVL    R0, STATUS                                       ;
                                      38          5A  E8 0010F          BLBS    STATUS, 12$                                      ; 1488
                                                  5A  DD 00112 10$:     PUSHL   STATUS                                           ; 1490
                               0000V  CF          01  FB 00114          CALLS   #1, COPY$MSG_NUMBER                              ;
           7E           00            50          01  7A 00119          EMUL    #1, R0, #0, -(SP)                                ;
           50           50            8E          08  7B 0011E          EDIV    #8, (SP)+, R0, R0                                ;
                                      04          50  D1 00123          CMPL    R0, #4                                           ;
                                                  12  13 00126          BEQL    11$                                              ;
                                                  5A  DD 00128          PUSHL   STATUS                                           ;
                               0000V  CF          01  FB 0012A          CALLS   #1, COPY$MSG_NUMBER                              ;
                                                  50  DD 0012F          PUSHL   R0                                               ;
                           00000000G  00          01  FB 00131          CALLS   #1, LIB$SIGNAL                                   ;
                                                  10  11 00138          BRB     12$                                              ;
                                                  5A  DD 0013A 11$:     PUSHL   STATUS                                           ;
                               0000V  CF          01  FB 0013C          CALLS   #1, COPY$MSG_NUMBER                              ;
                                                  50  DD 00141          PUSHL   R0                                               ;
                           00000000G  00          01  FB 00143          CALLS   #1, LIB$STOP                                     ;
                                      50          5A  D0 0014A 12$:     MOVL    STATUS, R0                                       ; 1492
                                                      04 0014D          RET                                                      ; 1494

; Routine Size:  334 bytes,    Routine Base:  $CODE$ + 0271
```

```
1 ROUTINE RMS_SETUP =                                            ! RMS RAB setup routine
1
1 !++
1 ! FUNCTIONAL DESCRIPTION:
1 !
1 !       This routine performs all necessary setup of the input and output file RABs:
1 !
1 !               * determine if record-mode is required
1 !               * allocate I/O buffers
1 !               * connect the RABs to their respective FABs
1 !
1 ! FORMAL PARAMETERS:
1 !
1 !       None
1 !
1 ! IMPLICIT INPUTS:
1 !
1 !       EXTEND_OUTFILE - Indicates output file is being extended
1 !       IO_BUFFER_BASE - location of the I/O buffer pool
1 !       INFILE_FAB - Input file FAB
1 !       OUTFILE_FAB - Output file FAB
1 !       INFILE_XABs - Input file XABs
1 !
1 ! IMPLICIT OUTPUTS:
1 !
1 !       INFILE_RAB - Input file RAB completed and connected
1 !       OUTFILE_RAB - Output file RAB completed and connected
1 !       IO_BUFFER_BASE - Address of dynamic I/O buffer (1st call only)
1 !       BLOCK_IO_SIZE - Length of block I/O operations
1 !
1 ! COMPLETION CODES:
1 !
1 !       OK = normal completion
1 !       ERROR = RAB connect unsuccessful
1 !
1 ! SIDE EFFECTS:
1 !
1 !       None
1 !
1 !--
1
2     BEGIN
2
2     LOCAL
2         IN_DEVICE :  BLOCK[1,BYTE],                            ! Selected input and output
2         OUT_DEVICE :  BLOCK[1,BYTE],                           ! device characteristics
2
2         FORCE_REC_MODE,                                        ! Temporary record-mode I/O indicator
2         STATUS,                                                ! System service completion code
2         IO_BUFFER_LENGTH : INITIAL(max_io_length*2),           ! Size of I/O buffer pool
2         GETSVI_ITEM_LIST : $ITMLST_DECL(ITEMS=1);              ! Item list for $GETSYI call
2
2     MACRO                                                      ! IN_DEVICE and OUT_DEVICE bit definitions:
2         DISK = 0,0,1,0 %,                                      !    disk device
2         TAPE = 0,1,1,0 %;                                      !    tape device
2
2 !
```

```
! Allocate a maximum size I/O buffer pool on the 1st call to this routine.
!

    IF .io_buffer_base EQL 0
    THEN
        BEGIN

        !
        ! Allocate enough virtual memory for the I/O buffer pool.  It has to
        ! be large enough to hold two of the largest possible RMS transfers.
        ! ****** NOTE ****** If COPY is ever made callable, the allocation of
        ! the I/O buffer pool will have to be rewritten to be more efficient.
        !

        IF NOT (status = LIB$GET_VM (io_buffer_length, io_buffer_base))
        THEN
            PUT_MESSAGE( MSG$_BADLOGIC, 0, .STATUS, 0,  MSG$_ATPC, 1 );
        END;

!
! Extract some device information from the input and output file FABs.
!

    IN_DEVICE = 0;                                          ! Clear the input and output
    OUT_DEVICE = 0;                                         ! device characteristics.

    IN_DEVICE[DISK] =                                       ! Turn on the input file disk indicator
        .INFILE_FAB[$FAB_DEV(FOD)] AND                      ! if the input device is file-structured
        NOT .INFILE_FAB[$FAB_DEV(SQD)];                     ! and it is not a tape device.

    IN_DEVICE[TAPE] =                                       ! Turn on the input file tape indicator
        .INFILE_FAB[$FAB_DEV(SQD)];                         ! if the input device is a tape.

    OUT_DEVICE[DISK] =                                      ! Turn on the output file disk indicator
        .OUTFILE_FAB[$FAB_DEV(FOD)] AND                     ! if the output device is file-structured
        NOT .OUTFILE_FAB[$FAB_DEV(SQD)];                    ! and it is not a tape device.

    OUT_DEVICE[TAPE] =                                      ! Turn on the output file tape indicator
        .OUTFILE_FAB[$FAB_DEV(SQD)];                        ! if the output device is a tape.

!
! Determine whether the input and output files have compatible attributes. This
! check can only be done if both the input and output devices are the same kind
! and they are file structured. The check should not be done if either the
! input device or the output device is a network device.
!

    IF .in_device NEQ .out_device
                OR
       .in_device EQL 0

    THEN
        force_rec_mode = YES
    ELSE
        IF NOT(.infile_fab[$FAB_DEV(NET)]                   ! If neither input or output is network then
                OR
           .outfile_fab[$FAB_DEV(NET)])
```

```
; 1082          1609  2            AND                                          ! Compare the following input and output XAB fields:
; 1083          1610  3         (IN_NEQ_OUT(XAB$B_RFO) OR                       !    record format and file organization
; 1084          1611  3          IN_NEQ_OUT(XAB$B_ATR) OR                       !    record attributes
; 1085          1612  3          IN_NEQ_OUT(XAB$B_BKZ) OR                       !    bucket size
; 1086          1613  3          IN_NEQ_OUT(XAB$B_HSZ) OR                       !    fixed header size
; 1087          1614  4          (.OUTFILE_XABFHC[XAB$W_MRZ] NEQ 0 AND          !    maximum output record size (if any)
; 1088          1615  4           .OUTFILE_XABFHC[XAB$W_MRZ] LSS                !       and longest input record
; 1089          1616  3                .INFILE_XABFHC[XAB$W_LRL]))              !
; 1090          1617  2      THEN
; 1091          1618  3          BEGIN                                          ! If the input and output attributes are not identic
; 1092          1619  3          IF NOT .COPY$B_INCOMPAT                        ! and this message has not appeared yet
; 1093          1620  3          THEN                                           ! for this output file.
; 1094          1621  4              BEGIN
; 1095        P 1622  4              PUT_MESSAGE( MSG$_INCOMPAT, 2,             ! send the user a warning message
; 1096          1623  4                      IN_NAME_DESC, OUT_NAME_DESC );     !
; 1097          1624  4              COPY$B_INCOMPAT = TRUE;                    ! Set flag saying that message is out.
; 1098          1625  3              END;
; 1099          1626  3          FORCE_REC_MODE = YES;                          ! and force a record-mode copy.
; 1100          1627  3          END
; 1101          1628  2      ELSE
; 1102          1629  2          FORCE_REC_MODE = NO;                           ! Otherwise, turn the record-mode indicator off.
; 1103          1630  2
; 1104          1631  2  !
; 1105          1632  2  ! Initialize the input and output RABs.
; 1106          1633  2  !
; 1107          1634  2
; 1108        P 1635  2    $RAB_INIT( RAB = INFILE_RAB,                         ! Setup the input file RAB as follows:
; 1109        P 1636  2               RAC = SEQ,                                !    Sequential record access
; 1110        P 1637  2               ROP = <LOC,RAH>,                          !    GET locate, read ahead
; 1111          1638  2               FAB = INFILE_FAB);                        !    Input file FAB address
; 1112          1639  2
; 1113        P 1640  2    $RAB_INIT( RAB = OUTFILE_RAB,                        ! Setup the output file RAB as follows:
; 1114        P 1641  2               RAC = SEQ,                                !    Sequential access
; 1115        P 1642  2               FAB = OUTFILE_FAB,                        !    Output file FAB address
; 1116          1643  2               ROP = <TPT,WBH> );                        !    Force EOF on every write or put,
; 1117          1644  2                                                         !    and specify write behind for multi-buffering.
; 1118          1645  2
; 1119          1646  2  !
; 1120          1647  2  !
; 1121          1648  2  ! Determine whether record-mode I/O is required for this file copy operation.
; 1122          1649  2  ! At least one of the following conditions must be true for record mode
; 1123          1650  2  ! operations to be performed:
; 1124          1651  2  !           - the input and output attributes are incompatible,
; 1125          1652  2  !           - the output file is being extended,
; 1126          1653  2  !           - the input and output devices are not the same type,
; 1127          1654  2  !           - both devices are record mode devices,
; 1128          1655  2  !           - this is a tape-to-tape copy AND
; 1129          1656  2  !               the input and output blocksizes are not the same
; 1130          1657  2  !                        OR
; 1131          1658  2  !               one tape is mounted foreign and the other is ANSI.
; 1132          1659  2  !
; 1133          1660  2
; 1134          1661  2    IF .FORCE_REC_MODE
; 1135          1662  2           OR
; 1136          1663  2       .EXTEND_OUTFILE
; 1137          1664  2           OR
; 1138          1665  2       .IN_DEVICE NEQ .OUT_DEVICE
```

```
                   OR
               .IN_DEVICE EQL 0
                   OR
               (
               .INFILE_FAB [$FAB_DEV (SQD)]
                   AND
               (
                   ( .INFILE_FAB [FAB$W_BLS] NEQ .OUTFILE_FAB [FAB$W_BLS] )
                   OR
                   ( .INFILE_FAB [$FAB_DEV (FOR)] NEQ .OUTFILE_FAB [$FAB_DEV (FOR)] )
               )
               )
    !
    ! Record mode I/O setup.
    !

        THEN
            BEGIN

            !
            ! Indicate that record mode is required, block i/o will not be used for
            ! this file, and that the record operations will be synchronous.
            !
            record_mode = YES;
            infile_rab[RAB$V_BIO]  = NO;
            outfile_rab[RAB$V_BIO] = NO;
            infile_rab[RAB$V_ASY]  = NO;
            outfile_rab[RAB$V_ASY] = NO;

            !
            ! Determine the size of the user's buffer which is passed to RMS.
            ! If the input device is tape, then the user's buffer must be large
            ! enough to contain one complete tape block.  Otherwise, (the input
            ! device is not tape) use either the the maximum record size or the
            ! the longest record length for the size of the user's buffer, if they
            ! are specfied.  If none of the above cases are met, use the longest
            ! legal transfer size as the length of the user's buffer.
            !
            IF .in_device[tape]
            THEN
                infile_rab[RAB$W_USZ] = .infile_fab[FAB$W_BLS]
            ELSE
                IF .infile_xabfhc[XAB$W_MRZ] NEQ 0
                THEN
                    infile_rab[RAB$W_USZ] = .infile_xabfhc[XAB$W_MRZ]
                ELSE
                    IF .infile_xabfhc[XAB$W_LRL] NEQ 0
                    THEN
                        infile_rab[RAB$W_USZ] = .infile_xabfhc[XAB$W_LRL]
                    ELSE
                        infile_rab[RAB$W_USZ] = max_io_length;


            !
            ! Set up the user's buffer within the I/O buffer pool.  If the record
            ! format of the file is VFC, then allocate areas in the buffer pool
            ! for the fixed header and variable portions of the record.  Otherwise,
   ! just use the start of the I/O buffer pool as the start of the user's
   ! buffer.
   !
   IF .infile_fab[FAB$B_RFM] EQL FAB$C_VFC
   THEN
       BEGIN
       infile_rab[RAB$L_RHB]  = .io_buffer_base;
       outfile_rab[RAB$L_RHB] = .infile_rab[RAB$L_RHB];
       infile_rab[RAB$L_UBF]  = .io_buffer_base + .infile_xabfhc[XAB$B_HSZ];
       END
   ELSE
       infile_rab[RAB$L_UBF] = .io_buffer_base;

   !
   ! Determine the best multi-block count for copying the input file.  Use
   ! that MBC for both the input and output file RABs.
   !
   IF .infile_fab [FAB$W_BLS] GTR .outfile_fab [FAB$W_BLS]
   THEN

       ! The input device is tape or some other record oriented device.
       ! Have RMS allocate enough buffer space to hold a complete block.
       !
       infile_rab [RAB$B_MBC] = (.infile_fab [FAB$W_BLS] + 511) / disk_block_size
   ELSE
       IF .outfile_fab [FAB$W_BLS] NEQ 0
       THEN

           ! The output device is record oriented and its block size is
           ! larger than the input device's.  Therefore, RMS should
           ! allocate enough buffer space to hold a comlete block for the
           ! output device.
           !
           infile_rab [RAB$B_MBC] = (.outfile_fab [FAB$W_BLS] + 511) / disk_block_size
       ELSE

           ! This is either a disk to disk transfer or something else.
           ! Just use the system default.
           !
           infile_rab [RAB$B_MBC] = .rms_mbc;

   outfile_rab [RAB$B_MBC] = .infile_rab [RAB$B_MBC];

   !
   ! Have RMS set up two internal buffers, to speed up processing.
   !
   infile_rab [RAB$B_MBF]  = double_buffer;
   outfile_rab [RAB$B_MBF] = double_buffer;
   END
!
! Block mode I/O setup.
!

 ELSE
   BEGIN
```

```
                !
                ! Indicate that record mode is not desired and that block mode will be
                ! used for both input and output, and that reading and writing will be
                ! synchronous.  However, ASY will be set after the $CONNECT to avoid
                ! haveing to issue a $WAIT on the connect.
                !

                record_mode = NO;
                infile_rab[RAB$V_BIO] = YES;
                outfile_rab[RAB$V_BIO] = YES;
                infile_rab[RAB$V_ASY] = NO;
                outfile_rab[RAB$V_ASY] = NO;

                !
                ! Determine the appropriate block size and user buffer size for copying
                ! the current input file.
                !
                IF .in_device[tape]
                THEN
                    BEGIN
                    block_size = .infile_fab [FAB$W_BLS];
                    infile_rab[RAB$W_USZ] = .infile_fab[FAB$W_BLS];
                    END
                ELSE
                    BEGIN
                    block_size = disk_block_size;
                    infile_rab[RAB$W_USZ] = .rms_mbc * disk_block_size;
                    END;

                !
                ! Set up the user's buffer, which are passed to RMS, within the I/O
                ! buffer pool.
                !

                infile_rab[RAB$L_UBF]  = .io_buffer_base;
                outfile_rab[RAB$L_RBF] = .io_buffer_base + .infile_rab[RAB$W_USZ];
                END;
        !
        ! Connect the input and output RABs to their respective FABs.
        !

            IF NOT $RMS_CONNECT( RAB = INFILE_RAB,                  ! Connect the input file RAB to the FAB,
                                 ERR = COPY$INOPN_ERR )             ! specifying an error action routine.

            THEN                                                    ! If the connect was not successful,
                RETURN NO_FILE;                                     ! return an error indication to the caller.

            IF .EXTEND_OUTFILE                                      ! If the output file is being extended,
            THEN
                OUTFILE_RAB[RAB$V_EOF] = YES;                       ! force end-of-file positioning on the following CON

            IF NOT $RMS_CONNECT( RAB = OUTFILE_RAB,                 ! Connect the output file RAB to the FAB,
                                 ERR = COPY$OUTOPN_ERR )            ! specifying an error action routine.

            THEN                                                    ! If the connect was not successful,
                RETURN NO_FILE;                                     ! return an error indication to the caller.
```

```
; 1310          1837  2
; 1311          1838  2 !
; 1312          1839  2 ! Set ASY bit in ROP if block I/O mode.
; 1313          1840  2 !
; 1314          1841  2
; 1315          1842  2    IF NOT .RECORD_MODE                                  ! If block I/O mode
; 1316          1843  2    THEN                                                 !
; 1317          1844  3        BEGIN                                            !
; 1318          1845  3        INFILE_RAB[RAB$V_ASY] = YES;                     ! indicate that reading and
; 1319          1846  3        OUTFILE_RAB[RAB$V_ASY] = YES;                    ! writing will be asynchronous
; 1320          1847  2        END;                                             !
; 1321          1848  2
; 1322          1849  2 !
; 1323          1850  2 ! Return to the caller
; 1324          1851  2 !
; 1325          1852  2
; 1326          1853  2    RETURN OK;                                           ! Return a success code to the caller.
; 1327          1854  1    END;


                                                              .EXTRN  SYS$CONNECT

                                        0FFC 00000 RMS_SETUP:
                                                              .WORD   Save R2,R3,R4,R5,R6,R7,R8,R9,R10,R11          ; 1495
                            5B     0000G  CF  9E 00002        MOVAB   INFILE_FAB+64, R11                            ;
                            5A     0000G  CF  9E 00007        MOVAB   $RMS_PTR+4, R10                               ;
                            59     0000G  CF  9E 0000C        MOVAB   $RMS_PTR+4, R9                                ;
                            5E            10  C2 00011        SUBL2   #16, SP                                       ;
                              0001FFFE    8F  DD 00014        PUSHL   #131070                                       ; 1536
                                   0000'  CF  D5 0001A        TSTL    IO_BUFFER_BASE                                ; 1555
                                          31  12 0001E        BNEQ    1$                                            ;
                                   0000'  CF  9F 00020        PUSHAB  IO_BUFFER_BASE                                ; 1566
                                     04   AE  9F 00024        PUSHAB  IO_BUFFER_LENGTH                              ;
                 00000000G  00            02  FB 00027        CALLS   #2, LIB$GET_VM                                ;
                            20            50  E8 0002E        BLBS    STATUS, 1$                                    ;
                                          01  DD 00031        PUSHL   #1                                            ; 1568
                            7E     115A   8F  3C 00033        MOVZWL  #4442, -(SP)                                  ;
                                          7E  D4 00038        CLRL    -(SP)                                         ;
                                          50  DD 0003A        PUSHL   STATUS                                        ;
                                          7E  D4 0003C        CLRL    -(SP)                                         ;
                            7E     1124   8F  3C 0003E        MOVZWL  #4388, -(SP)                                  ;
                     0000V  CF            01  FB 00043        CALLS   #1, COPY$MSG_NUMBER                           ;
                                          50  DD 00048        PUSHL   R0                                            ;
                 00000000G  00            06  FB 0004A        CALLS   #6, LIB$STOP                                  ;
                                          57  94 00051 1$:    CLRB    IN_DEVICE                                     ; 1575
                                          50  94 00053        CLRB    OUT_DEVICE                                    ; 1576
   51       01   AB         01            06  EF 00055        EXTZV   #6, #1, INFILE_FAB+65, R1                     ; 1580
   52            6B         01            05  EF 0005B        EXTZV   #5, #1, INFILE_FAB+64, R2                     ;
                            51            52  8A 00060        BICB2   R2, R1                                        ;
   57            01         00            51  F0 00063        INSV    R1, #0, #1, IN_DEVICE                         ;
   51            6B         01            05  EF 00068        EXTZV   #5, #1, INFILE_FAB+64, R1                     ; 1583
   57            01         01            51  F0 0006D        INSV    R1, #1, #1, IN_DEVICE                         ;
   51     0000G  CF         01            06  EF 00072        EXTZV   #6, #1, OUTFILE_FAB+65, R1                    ; 1587
   52     0000G  CF         01            05  EF 00079        EXTZV   #5, #1, OUTFILE_FAB+64, R2                    ;
                            51            52  8A 00080        BICB2   R2, R1                                        ;
   50            01         00            51  F0 00083        INSV    R1, #0, #1, OUT_DEVICE                        ;
```

```
                                                              G 9
COPYMAIN                                              15-Sep-1984 23:39:26    VAX-11 Bliss-32 V4.0-742               Page  34
V04-000                                               14-Sep-1984 12:14:18    [COPY.SRC]COPYMAIN.B32;1                     (8)

           51    0000G  CF              01             05  EF  00088        EXTZV   #5, #1, OUTFILE_FAB+64, R1              ; 1590
           50           01              01             51  F0  0008F        INSV    R1, #1, #1, OUT_DEVICE                  ;
                                                       58  D4  00094        CLRL    R8                                      ; 1599
                                        50             57  91  00096        CMPB    IN_DEVICE, OUT_DEVICE                   ;
                                                       04  13  00099        BEQL    2$                                      ;
                                                       58  D6  0009B        INCL    R8                                      ;
                                                       68  11  0009D        BRB     4$                                      ;
                                                       57  95  0009F  2$:   TSTB    IN_DEVICE                               ; 1601
                                                       64  13  000A1        BEQL    4$                                      ;
                        64       01     AB             05  E0  000A3        BBS     #5, INFILE_FAB+65, 5$                   ; 1606
                        5E    0000G     CF             05  E0  000A8        BBS     #5, OUTFILE_FAB+65, 5$                  ; 1608
                              0000G     CF     0000G   CF  91  000AE        CMPB    INFILE_XABFHC+8, OUTFILE_XABFHC+8       ; 1610
                                                       29  12  000B5        BNEQ    3$                                      ;
                              0000G     CF     0000G   CF  91  000B7        CMPB    INFILE_XABFHC+9, OUTFILE_XABFHC+9       ; 1611
                                                       20  12  000BE        BNEQ    3$                                      ;
                              0000G     CF     0000G   CF  91  000C0        CMPB    INFILE_XABFHC+22, OUTFILE_XABFHC+22     ; 1612
                                                       17  12  000C7        BNEQ    3$                                      ;
                              0000G     CF     0000G   CF  91  000C9        CMPB    INFILE_XABFHC+23, OUTFILE_XABFHC+23     ; 1613
                                                       0E  12  000D0        BNEQ    3$                                      ;
                                        50     0000G   CF  3C  000D2        MOVZWL  OUTFILE_XABFHC+24, R0                   ; 1614
                                                       33  13  000D7        BEQL    5$                                      ;
                                        50     0000G   CF  B1  000D9        CMPW    INFILE_XABFHC+10, R0                    ; 1616
                                                       2C  1B  000DE        BLEQU   5$                                      ;
                                        22     0000'   CF  E8  000E0  3$:   BLBS    COPY$B_INCOMPAT, 4$                     ; 1619
                                               0000G   CF  9F  000E5        PUSHAB  OUT_NAME_DESC                           ; 1623
                                               0000G   CF  9F  000E9        PUSHAB  IN_NAME_DESC                            ;
                                                       02  DD  000ED        PUSHL   #2                                      ;
                                        7E      11E0   8F  3C  000EF        MOVZWL  #4576, -(SP)                            ;
                              0000V     CF             01  FB  000F4        CALLS   #1, COPY$MSG_NUMBER                     ;
                                                       50  DD  000F9        PUSHL   R0                                      ;
                          000000000G    00             04  FB  000FB        CALLS   #4, LIB$SIGNAL                          ;
                              0000'     CF             01  90  00102        MOVB    #1, COPY$B_INCOMPAT                     ; 1624
                                        56             01  D0  00107  4$:   MOVL    #1, FORCE_REC_MODE                      ; 1626
                                                       02  11  0010A        BRB     6$                                      ; 1606
                                                       56  D4  0010C  5$:   CLRL    FORCE_REC_MODE                          ; 1629
    0044   8F           00              6E             00  2C  0010E  6$:   MOVC5   #0, (SP), #0, #68, $RMS_PTR             ; 1638
                                                FC     A9      00115                                                    ;
                                FC      A9      4401   8F  B0  00117        MOVW    #17409, $RMS_PTR                        ;
                                        69  00010200   8F  D0  0011D        MOVL    #66048, $RMS_PTR+4                      ;
                                                1A     A9  94  00124        CLRB    $RMS_PTR+30                             ;
                                38      A9      C0     AB  9E  00127        MOVAB   INFILE_FAB, $RMS_PTR+60                 ;
    0044   8F           00              6E             00  2C  0012C        MOVC5   #0, (SP), #0, #68, $RMS_PTR             ; 1643
                                                FC     AA      00133                                                    ;
                                FC      AA      4401   8F  B0  00135        MOVW    #17409, $RMS_PTR                        ;
                                        6A      0402   8F  3C  0013B        MOVZWL  #1026, $RMS_PTR+4                       ;
                                                1A     AA  94  00140        CLRB    $RMS_PTR+30                             ;
                                38      AA     0000G   CF  9E  00143        MOVAB   OUTFILE_FAB, $RMS_PTR+60                ;
                                        51     0000'   CF  D0  00149        MOVL    IO_BUFFER_BASE, R1                      ; 1729
                                        26             56  E8  0014E        BLBS    FORCE_REC_MODE, 9$                      ; 1661
                                               0000'   CF  95  00151        TSTB    COPY$SEM_STATUS+2                       ; 1663
                                                       20  19  00155        BLSS    9$                                      ;
                                        1D             58  E8  00157        BLBS    R8, 9$                                  ; 1665
                                                       57  95  0015A        TSTB    IN_DEVICE                               ; 1667
                                                       19  13  0015C        BEQL    9$                                      ;
                        03              6B             05  E0  0015E        BBS     #5, INFILE_FAB+64, 8$                   ; 1670
                                                     00B6  31  00162  7$:   BRW     19$                                     ;
                              0000G     CF      FC     AB  B1  00165  8$:   CMPW    INFILE_FAB+60, OUTFILE_FAB+60           ; 1673
```

```
                                       0A   12 0016B         BNEQ    9$                                                 ;
            50   0000G  CF      03     AB   8D 0016D         XORB3   INFILE_FAB+67, OUTFILE_FAB+67, R0                  ; 1675
                        EB             50   E9 00174         BLBC    R0, 7$                                             ;
                 0000'  CF      40     8F   88 00177 9$:     BISB2   #64, COPY$SEM_STATUS+2                             ; 1689
                   01   A9             08   8A 0017D         BICB2   #8, INFILE_RAB+5                                   ; 1690
                   01   AA             08   8A 00181         BICB2   #8, OUTFILE_RAB+5                                  ; 1691
                        69             01   8A 00185         BICB2   #1, INFILE_RAB+4                                   ; 1692
                        6A             01   8A 00188         BICB2   #1, OUTFILE_RAB+4                                  ; 1693
            07          57             01   E1 0018B         BBC     #1, IN_DEVICE, 10$                                 ; 1704
                   1C   A9      FC     AB   B0 0018F         MOVW    INFILE_FAB+60, INFILE_RAB+32                       ; 1706
                                       18   11 00194         BRB     13$                                                ;
                        50   0000G     CF   3C 00196 10$:    MOVZWL  INFILE_XABFHC+24, R0                               ; 1708
                                       07   12 0019B         BNEQ    11$                                                ;
                        50   0000G     CF   3C 0019D         MOVZWL  INFILE_XABFHC+10, R0                               ; 1712
                                       06   13 001A2         BEQL    12$                                                ;
                   1C   A9             50   B0 001A4 11$:    MOVW    R0, INFILE_RAB+32                                  ; 1714
                                       04   11 001A8         BRB     13$                                                ;
                   1C   A9             01   AE 001AA 12$:    MNEGW   #1, INFILE_RAB+32                                  ; 1716
                        03      DF     AB   91 001AE 13$:    CMPB    INFILE_FAB+31, #3                                  ; 1726
                                       15   12 001B2         BNEQ    14$                                                ;
                   28   A9             51   D0 001B4         MOVL    R1, INFILE_RAB+44                                  ; 1729
                   28   AA      28     A9   D0 001B8         MOVL    INFILE_RAB+44, OUTFILE_RAB+44                      ; 1730
                   50        0000G     CF   9A 001BD         MOVZBL  INFILE_XABFHC+23, R0                               ; 1731
      20    A9     51                  50   C1 001C2         ADDL3   R0, R1, INFILE_RAB+36                              ;
                                       04   11 001C7         BRB     15$                                                ; 1726
                   20   A9             51   D0 001C9 14$:    MOVL    R1, INFILE_RAB+36                                  ; 1734
                        50   0000G     CF   3C 001CD 15$:    MOVZWL  OUTFILE_FAB+60, R0                                 ; 1740
                        50      FC     AB   B1 001D2         CMPW    INFILE_FAB+60, R0                                  ;
                                       17   1B 001D6         BLEQU   16$                                                ;
                        51      FC     AB   3C 001D8         MOVZWL  INFILE_FAB+60, R1                                  ; 1746
                        51    01FF     C1   9E 001DC         MOVAB   511(R1), R1                                        ;
            52          51 00000200    8F   C7 001E1         DIVL3   #512, R1, R2                                       ;
                   33   A9             52   90 001E9         MOVB    R2, INFILE_RAB+55                                  ;
                                       1D   11 001ED         BRB     18$                                                ;
                                       50   D5 001EF 16$:    TSTL    R0                                                 ; 1748
                                       13   13 001F1         BEQL    17$                                                ;
                        50    01FF     C0   9E 001F3         MOVAB   511(R0), R0                                        ; 1756
            51          50 00000200    8F   C7 001F8         DIVL3   #512, R0, R1                                       ;
                   33   A9             51   90 00200         MOVB    R1, INFILE_RAB+55                                  ;
                                       06   11 00204         BRB     18$                                                ;
                   33   A9    0000'    CF   90 00206 17$:    MOVB    RMS_MBC, INFILE_RAB+55                             ; 1762
                   33   AA      33     A9   90 0020C 18$:    MOVB    INFILE_RAB+55, OUTFILE_RAB+55                      ; 1764
                   32   A9             02   90 00211         MOVB    #2, INFILE_RAB+54                                  ; 1769
                   32   AA             02   90 00215         MOVB    #2, OUTFILE_RAB+54                                 ; 1770
                                       42   11 00219         BRB     22$                                                ; 1661
                 0000'  CF      40     8F   8A 0021B 19$:    BICB2   #64, COPY$SEM_STATUS+2                             ; 1787
                   01   A9             08   88 00221         BISB2   #8, INFILE_RAB+5                                   ; 1788
                   01   AA             08   88 00225         BISB2   #8, OUTFILE_RAB+5                                  ; 1789
                        69             01   8A 00229         BICB2   #1, INFILE_RAB+4                                   ; 1790
                        6A             01   8A 0022C         BICB2   #1, OUTFILE_RAB+4                                  ; 1791
            0D          57             01   E1 0022F         BBC     #1, IN_DEVICE, 20$                                 ; 1797
                 0000'  CF      FC     AB   3C 00233         MOVZWL  INFILE_FAB+60, BLOCK_SIZE                          ; 1800
                   1C   A9      FC     AB   B0 00239         MOVW    INFILE_FAB+60, INFILE_RAB+32                       ; 1801
                                       10   11 0023E         BRB     21$                                                ; 1797
                 0000'  CF    0200     8F   3C 00240 20$:    MOVZWL  #512, BLOCK_SIZE                                   ; 1805
      1C    A9   0000'  CF    0200     8F   A5 00247         MULW3   #512, RMS_MBC, INFILE_RAB+32                       ; 1806
                   20   A9             51   D0 00250 21$:    MOVL    R1, INFILE_RAB+36                                  ; 1814
```

```
                                 50       1C   A9  3C 00254         MOVZWL  INFILE_RAB+32, R0                            ; 1815
                24   AA          51            50  C1 00258         ADDL3   R0, R1, OUTFILE_RAB+40                       ;
                                       0000V   CF  9F 0025D 22$:    PUSHAB  COPY$INOPN_ERR                               ; 1823
                                          FC   A9  9F 00261         PUSHAB  INFILE_RAB                                   ;
                        00000000G  00          02  FB 00264         CALLS   #2, SYS$CONNECT                              ;
                                   2B          50  E9 0026B         BLBC    R0, 25$                                      ;
                                       0000'   CF  95 0026E         TSTB    COPY$SEM_STATUS+2                            ; 1828
                                               04  18 00272         BGEQ    23$                                          ;
                             01   AA           01  88 00274         BISB2   #1, OUTFILE_RAB+5                            ; 1830
                                       0000V   CF  9F 00278 23$:    PUSHAB  COPY$OUTOPN_ERR                              ; 1833
                                          FC   AA  9F 0027C         PUSHAB  OUTFILE_RAB                                  ;
                        00000000G  00          02  FB 0027F         CALLS   #2, SYS$CONNECT                              ;
                                   10          50  E9 00286         BLBC    R0, 25$                                      ;
                06       0000'     CF          06  E0 00289         BBS     #6, COPY$SEM_STATUS+2, 24$                   ; 1842
                                   69          01  88 0028F         BISB2   #1, INFILE_RAB+4                             ; 1845
                                   6A          01  88 00292         BISB2   #1, OUTFILE_RAB+4                            ; 1846
                                   50          01  D0 00295 24$:    MOVL    #1, R0                                       ; 1853
                                                   04 00298         RET                                                  ;
                                               50  D4 00299 25$:    CLRL    R0                                           ; 1854
                                                   04 0029B         RET                                                  ;

; Routine Size:  668 bytes,    Routine Base:  $CODE$ + 03BF
```

```
1 PSECT CODE = COPY$COPY_FILE (ALIGN(9));                    ! Force page alignment for this routine.
1
1 ROUTINE COPY_FILE =                                        ! Copies an entire input file to the output file
1
1 !++
1 ! FUNCTIONAL DESCRIPTION:
1 !
1 !       This routine copies an entire input file into the output file,
1 !       using block mode I/O if possible.
1 !
1 !       This routine is page-aligned in order to minimize page faulting
1 !       due to executing the code which performs the actual file copying.
1 !
1 ! FORMAL PARAMETERS:
1 !
1 !       None
1 !
1 ! IMPLICIT INPUTS:
1 !
1 !       RECORD_MODE - Indicates whether record mode I/O is required
1 !       INFILE_FAB - Input file FAB
1 !       INFILE_RAB - Input file RAB
1 !
1 ! IMPLICIT OUTPUTS:
1 !
1 !       RECORD_COUNT - Number of input file records copied
1 !       BLOCK_COUNT - Number of input file blocks copied
1 !
1 ! COMPLETION CODES:
1 !
1 !       OK = successful copy
1 !       ERROR = I/O error during copy
1 !
1 ! SIDE EFFECTS:
1 !
1 !       None
1 !
1 !--
1
2    BEGIN
2
2    LOCAL
2       NEXT_READ;                                           ! Temporary buffer pointer
2
2 !
2 ! Initialization
2 !
2
2    RECORD_COUNT = 0;                                       ! Zero the input file record
2    BLOCK_COUNT = 0;                                        ! and block counters.
2
2 !
2 ! If necessary, copy the input file to the output file one record at a time.
2 !
2
2    IF .RECORD_MODE                                         ! Test the record mode I/O indicator.
2    THEN
```

```
2           WHILE 1 DO                                      ! Beginning of the record copying loop which
3               BEGIN                                       ! will be terminated by a RETURN in the event
3                                                           ! of an input end-of-file or any I/O error.
3
4               IF NOT $RMS_GET( RAB = INFILE_RAB )         ! Get one record from the input file.
4
3               THEN                                        ! If the get was not successful,
4                   BEGIN                                   ! begin error processing.
4                                                           !
4                   IF .INFILE_RAB[RAB$L_STS] EQL RMS$_EOF  ! If the error was an input end-of-file,
4                   THEN                                    !
4                       RETURN OK;                          ! return a success code to the caller.
4
4                   IN_READ_ERROR();                        ! Otherwise, send an error message to the user
4                   RETURN ERROR;                           ! and return an error code to the caller.
4
3                   END;                                    ! End of input error processing.
3
3               OUTFILE_RAB[RAB$L_RBF] =                    ! Copy the input record address
3                           .INFILE_RAB[RAB$L_RBF];         ! and record length
3               OUTFILE_RAB[RAB$W_RSZ] =                    ! from the input file RAB
3                           .INFILE_RAB[RAB$W_RSZ];         ! to the output file RAB.
3
4               IF $RMS_PUT( RAB = OUTFILE_RAB )            ! Write one record into the output file.
4
3               THEN                                        ! If the put was successful,
3                   RECORD_COUNT = .RECORD_COUNT + 1        ! increment the record counter.
3               ELSE
4                   BEGIN                                   ! Otherwise,
4                   OUT_WRITE_ERROR();                      ! send an error message to the user
4                   RETURN ERROR;                           ! and return to the caller.
3                   END;
3
3               END                                         ! End of record mode copy loop.
3   !
3   ! If possible, copy the input file to the output file a block at a time.
3   !
3
2       ELSE
2           WHILE 1 DO                                      ! Beginning of the block copying loop which
3               BEGIN                                       ! will be terminated by a RETURN in the event
3                                                           ! of an input end-of-file or any I/O error.
3
3               $RMS_READ( RAB = INFILE_RAB );              ! Begin an asynchronous read from the input file.
3
4               IF NOT $RMS_WAIT( RAB = OUTFILE_RAB )       ! Wait for the previous write to complete.
3               THEN
4                   BEGIN                                   ! If the write was not successful,
4                   OUT_WRITE_ERROR();                      ! send the user an error message,
4                   $RMS_WAIT( RAB = INFILE_RAB );          ! wait for the previous read to complete,
4                   RETURN ERROR;                           ! and then return an error code to the caller.
3                   END;
3
4               IF $RMS_WAIT( RAB = INFILE_RAB )            ! Wait for the previous read to complete.
3               THEN
4                   BEGIN                                   ! If the read was successful,
; 1443          1969  4             INFILE_RAB[RAB$L_UBF] =                     ! save the current output buffer address
; 1444          1970  4                   .OUTFILE_RAB[RAB$L_RBF];              !
; 1445          1971  4             OUTFILE_RAB[RAB$L_RBF] =                    ! and copy the input block address and block size
; 1446          1972  4                   .INFILE_RAB[RAB$L_RBF];               ! from the input file RAB into the output RAB.
; 1447          1973  4             OUTFILE_RAB[RAB$W_RSZ] =                    !
; 1448          1974  4                   .INFILE_RAB[RAB$W_RSZ];               !
; 1449          1975  4
; 1450          1976  4             $RMS_WRITE( RAB = OUTFILE_RAB );            ! Initiate an asynchronous write.
; 1451          1977  4
; 1452          1978  4             BLOCK_COUNT = .BLOCK_COUNT +                ! Increment the count of blocks written.
; 1453          1979  5                 (.INFILE_RAB[RAB$W_RSZ] +               !
; 1454          1980  4                 .BLOCK_SIZE - 1) / .BLOCK_SIZE;         !
; 1455          1981  4             END
; 1456          1982  3         ELSE                                            ! If the read was unsuccessful,
; 1457          1983  4             BEGIN                                       ! begin special input error processing.
; 1458          1984  4
; 1459          1985  4             IF .INFILE_RAB[RAB$L_STS] EQL RMS$_EOF      ! If the error was an input end-of-file,
; 1460          1986  4             THEN                                        !
; 1461          1987  4                 RETURN OK;                              ! return a success code to the caller.
; 1462          1988  4
; 1463          1989  4             IN_READ_ERROR();                            ! Otherwise, send an error message to the user
; 1464          1990  4             RETURN ERROR;                               ! and then return an error code to the caller.
; 1465          1991  3             END;
; 1466          1992  3
; 1467          1993  2         END;                                            ! End of block mode copy loop.
; 1468          1994  2
; 1469          1995  2     RETURN OK;
; 1470          1996  1     END;




                                                                 .EXTRN  SYS$GET, SYS$PUT
                                                                 .EXTRN  SYS$READ, SYS$WAIT
                                                                 .EXTRN  SYS$WRITE

                                                                 .PSECT  COPY$COPY_FILE,NOWRT,9

                                        003C 00000 COPY_FILE:
                                                                 .WORD   Save R2,R3,R4,R5                           ; 1857
                     55       0000'  CF  9E 00002                MOVAB   BLOCK_COUNT, R5                            ;
                     54  00000000G   00  9E 00007                MOVAB   SYS$WAIT, R4                               ;
                     53       0000G  CF  9E 0000E                MOVAB   OUTFILE_RAB+40, R3                         ;
                     52       0000G  CF  9E 00013                MOVAB   INFILE_RAB, R2                             ;
                                     65  7C 00018                CLRQ    BLOCK_COUNT                                ; 1904
          2E      36  A5             06  E1 0001A                BBC     #6, COPY$SEM_STATUS+2, 3$                  ; 1910
                                     52  DD 0001F 1$:            PUSHL   R2                                         ; 1916
         00000000G    00             01  FB 00021                CALLS   #1, SYS$GET                                ;
                      72             50  E9 00028                BLBC    R0, 5$                                     ;
                      63         28  A2  D0 0002B                MOVL    INFILE_RAB+40, OUTFILE_RAB+40              ; 1931
                  FA  A3         22  A2  B0 0002F                MOVW    INFILE_RAB+34, OUTFILE_RAB+34              ; 1933
                                 D8  A3  9F 00034                PUSHAB  OUTFILE_RAB                                ; 1935
         00000000G    00             01  FB 00037                CALLS   #1, SYS$PUT                                ;
                      05             50  E9 0003E                BLBC    R0, 2$                                     ;
                                 04  A5  D6 00041                INCL    RECORD_COUNT                               ; 1938
                                     D9  11 00044                BRB     1$                                         ;
             0000V  CF               00  FB 00046 2$:            CALLS   #0, OUT_WRITE_ERROR                        ; 1941
                                     5F  11 0004B                BRB     6$                                         ; 1942
```

```
                                  52  DD 0004D 3$:    PUSHL    R2                                        ; 1956
               00000000G  00      01  FB 0004F        CALLS    #1, SYS$READ                              ;
                              D8  A3  9F 00056        PUSHAB   OUTFILE_RAB                               ; 1958
                          64      01  FB 00059        CALLS    #1, SYS$WAIT                              ;
                          0C      50  E8 0005C        BLBS     R0, 4$                                    ;
                   0000V  CF      00  FB 0005F        CALLS    #0, OUT_WRITE_ERROR                       ; 1961
                                  52  DD 00064        PUSHL    R2                                        ; 1962
                          64      01  FB 00066        CALLS    #1, SYS$WAIT                              ;
                                  41  11 00069        BRB      6$                                        ; 1963
                                  52  DD 0006B 4$:    PUSHL    R2                                        ; 1966
                          64      01  FB 0006D        CALLS    #1, SYS$WAIT                              ;
                          2A      50  E9 00070        BLBC     R0, 5$                                    ;
                      24  A2      63  D0 00073        MOVL     OUTFILE_RAB+40, INFILE_RAB+36             ; 1970
                          63  28  A2  D0 00077        MOVL     INFILE_RAB+40, OUTFILE_RAB+40             ; 1972
                      FA  A3  22  A2  B0 0007B        MOVW     INFILE_RAB+34, OUTFILE_RAB+34             ; 1974
                              D8  A3  9F 00080        PUSHAB   OUTFILE_RAB                               ; 1976
               00000000G  00      01  FB 00083        CALLS    #1, SYS$WRITE                             ;
                          50  22  A2  3C 0008A        MOVZWL   INFILE_RAB+34, R0                         ; 1980
                          50  14  A5  C0 0008E        ADDL2    BLOCK_SIZE, R0                            ;
                                  50  D7 00092        DECL     R0                                        ; 1979
                          50  14  A5  C6 00094        DIVL2    BLOCK_SIZE, R0                            ; 1980
                          65      50  C0 00098        ADDL2    R0, BLOCK_COUNT                           ;
                                  B0  11 0009B        BRB      3$                                        ; 1966
               0001827A   8F  08  A2  D1 0009D 5$:    CMPL     INFILE_RAB+8, #98938                      ; 1985
                                  09  13 000A5        BEQL     7$                                        ;
                   0000V  CF      00  FB 000A7        CALLS    #0, IN_READ_ERROR                         ; 1989
                          50      02  D0 000AC 6$:    MOVL     #2, R0                                    ; 1990
                                      04 000AF        RET                                                ;
                          50      01  D0 000B0 7$:    MOVL     #1, R0                                    ; 1995
                                      04 000B3        RET                                                ; 1996

; Routine Size:  180 bytes,    Routine Base:  COPY$COPY_FILE + 0000

; 1471          1997  1 PSECT CODE = $CODE$;                            ! Resume the default PSECT (see previous routine).
```

```
  1 ROUTINE CLOSE_INFILE :  NOVALUE =                             ! Close the current input file
  1
  1 !++
  1 ! FUNCTIONAL DESCRIPTION:
  1 !
  1 !       This routine closes the current input file.
  1 !
  1 ! FORMAL PARAMETERS:
  1 !
  1 !       None
  1 !
  1 ! IMPLICIT INPUTS:
  1 !
  1 !       INFILE_OPEN - Input file open indicator
  1 !       INFILE_FAB - Input file FAB
  1 !
  1 ! IMPLICIT OUTPUTS:
  1 !
  1 !       INFILE_OPEN - Set to indicate that the input file is not open
  1 !       INFILE_FAB - Input file FAB closed
  1 !
  1 ! ROUTINE VALUE:
  1 !
  1 !       None
  1 !
  1 ! SIDE EFFECTS:
  1 !
  1 !       None
  1 !
  1 !--
  1
  2     BEGIN
  2
  2 !
  2 ! Return to the caller if the input file is not open.
  2 !
  2
  2     IF NOT .INFILE_OPEN                                       ! If the input file is not open,
  2     THEN                                                      !
  2         RETURN;                                               ! return to the caller.
  2
  2     INFILE_OPEN = NO;                                         ! Otherwise, turn off the open indicator.
  2
  2 !
  2 ! Close the input file.
  2 !
  2
P 2     $RMS_CLOSE( FAB = INFILE_FAB,                             ! Close the input file FAB,
  2                 ERR = IN_CLOSE_ERROR );                       ! specifying an error action routine.
  2
  2 !
  2 ! Return to the caller.
  2 !
  2
  2     RETURN;                                                   ! Return to the caller.
  2
  1     END;
```

```
                                                                         .EXTRN  SYS$CLOSE

                                                                         .PSECT  $CODE$,NOWRT,2

                                                      0000 00000 CLOSE_INFILE:
                                                                         .WORD   Save nothing                                  ; 1998
                       14     0000'  CF              02  E1 00002        BBC     #2, COPY$SEM_STATUS+2, 1$                     ; 2035
                              0000'  CF              04  8A 00008        BICB2   #4, COPY$SEM_STATUS+2                         ; 2039
                                        0000V  CF    9F 0000D           PUSHAB  IN_CLOSE_ERROR                                ; 2046
                                        0000G  CF    9F 00011           PUSHAB  INFILE_FAB                                    ;
                         00000000G  00              02  FB 00015        CALLS   #2, SYS$CLOSE                                 ;
                                                         04 0001C 1$:    RET                                                   ; 2054

; Routine Size:  29 bytes,    Routine Base:  $CODE$ + 065B
```

```
  1 GLOBAL ROUTINE COPY$CLOSE_OUTF :  NOVALUE =           ! Close the current output file
  1
  1 !++
  1 ! FUNCTIONAL DESCRIPTION:
  1 !
  1 !       This routine closes the current output file.
  1 !
  1 ! FORMAL PARAMETERS:
  1 !
  1 !       None
  1 !
  1 ! IMPLICIT INPUTS:
  1 !
  1 !       OUTFILE_OPEN - Output file open indicator
  1 !       OUTFILE_FAB - Output file FAB
  1 !       TRUNCATE_BIT in COPY$CLI_STATUS if /TRUNCATE was specified.
  1 !
  1 ! IMPLICIT OUTPUTS:
  1 !
  1 !       OUTFILE_OPEN - Set to indicate that the output file is not open
  1 !       OUTFILE_FAB - Output file FAB closed
  1 !
  1 ! ROUTINE VALUE:
  1 !
  1 !       None
  1 !
  1 ! SIDE EFFECTS:
  1 !
  1 !       File is truncated if /TRUNCATE was specified.
  1 !
  1 !--
  1
  2    BEGIN
  2
  2 !
  2 ! Return to the caller if the output file is not open.
  2 !
  2
  2    IF NOT .OUTFILE_OPEN                               ! If the output file is not open,
  2    THEN                                               !
  2        RETURN OK;                                     ! return a success code to the caller.
  2
  2    OUTFILE_OPEN = NO;                                 ! Otherwise, turn off the open indicator.
  2
  2 !
  2 ! Close the output file.
  2 !
  2
P 2    $RMS_CLOSE( FAB = OUTFILE_FAB,                     ! Close the output file FAB,
  2                ERR = COPY$OCLOSE_ERR );               ! specifying an error action routine.
  2 !
  2 ! Reset the incompatible messages flag to FALSE for the next output file. This message
  2 ! indicates whether an incompatible attributes has been output for an output file.
  2 !
  2
  2    COPY$B_INCOMPAT = FALSE;                           ! Reset incompatible flag
```

```
; 1588          2112  2
; 1589          2113  2  !
; 1590          2114  2  ! Return to the caller.
; 1591          2115  2  !
; 1592          2116  2
; 1593          2117  2     RETURN;                                             ! Return to the caller.
; 1594          2118  2
; 1595          2119  1     END;



                                       0000 00000        .ENTRY  COPY$CLOSE_OUTF, Save nothing                 ; 2055
           18      0000'  CF          01   E1 00002        BBC     #1, COPY$SEM_STATUS+2, 1$                     ; 2093
                   0000'  CF          02   8A 00008        BICB2   #2, COPY$SEM_STATUS+2                         ; 2097
                            0000V  CF   9F 0000D        PUSHAB  COPY$OCLOSE_ERR                               ; 2104
                            0000G  CF   9F 00011        PUSHAB  OUTFILE_FAB                                   ;
              00000000G  00           02   FB 00015        CALLS   #2, SYS$CLOSE                                 ;
                            0000'  CF   94 0001C        CLRB    COPY$B_INCOMPAT                               ; 2111
                                        04 00020 1$:    RET                                                   ; 2119

; Routine Size:  33 bytes,    Routine Base:  $CODE$ + 0678
```

```
ROUTINE BYPASS_CONCAT =                                         ! Bypass concatenated input files

!++
! FUNCTIONAL DESCRIPTION:
!
!       This routine scans past concatenated input file-specifications.
!
! FORMAL PARAMETERS:
!
!       None
!
! IMPLICIT INPUTS:
!
!       Bits in the status words COPY$CLI_STATUS and COPY$SEM_STATUS:
!
!               APPEND_COMMAND  - APPEND command indicator
!               CONCAT_FOLLOWS  - concatentation is occurring
!
!       INFILE_DESC - Input file request descriptor
!       CLEANUP_DESC - Input file cleanup request descriptor
!
! IMPLICIT OUTPUTS:
!
!       CONCAT_FOLLOWS - Concatenation active indicator turned off
!       WILDCARD_ACTIVE - Wildcard active indicator turned off
!
! ROUTINE VALUE:
!
!       None
!
! SIDE EFFECTS:
!
!       INFILE_DESC - Input file request descriptor filled in by the CLI
!       CLEANUP_DESC - Input file cleanup request descriptor filled in by the CLI
!
!--

    BEGIN

    LOCAL
        DESC : $BBLOCK[ DSC$C_S_BLN ]                           ! Descriptor for input file name
        ;

!
! Initialize descriptor.
!
    CH$FILL( 0, DSC$C_S_BLN, DESC);
    DESC[ DSC$B_CLASS ] = DSC$K_CLASS_D;

!
! Return to the caller if input concatenation is not active.
!

    IF NOT .APPEND_COMMAND AND                                  ! If this is a COPY command
        NOT .CONCAT_FOLLOWS                                     ! and no input concatentation is active,
    THEN                                                        !
        RETURN false                                                    !    then return to the caller.
```

```
; 1654          2177 2      ELSE
; 1655          2178 2          CONCAT_FOLLOWS = NO;                                ! Otherwise, turn off the concatenation indicator.
; 1656          2179 2
; 1657          2180 2    !
; 1658          2181 2    ! Report an wildcard specification which has not been completely processed.
; 1659          2182 2    !
; 1660          2183 2
; 1661          2184 2      IF .WILDCARD_ACTIVE                                     ! If a wildcard spec is currently active,
; 1662          2185 2      THEN                                                    !
; 1663          2186 3          BEGIN                                               !
; 1664          2187 3          WILDCARD_ACTIVE = NO;                               ! turn off the wildcard indicator.
; 1665          2188 3
; 1666          2189 3          IF .INFILE_NAM_BLK[NAM$B_RSL] NEQ 0                 ! If the wildcard spec is partially processed,
; 1667          2190 3          THEN                                                !
; 1668          2191 4              BEGIN                                           !
; 1669          2192 4              INFILE_NAM_BLK[NAM$B_RSL] = 0;                  ! discard the current resultant name string,
; 1670          2193 4              REPORT_BYPASS( MSG$_NOTCMPLT );                 ! and report the bypass wildcard spec.
; 1671          2194 3              END;
; 1672          2195 2          END;
; 1673          2196 2
; 1674          2197 2    !
; 1675          2198 2    ! Scan past any concatenated input file-specifications.
; 1676          2199 2    !
; 1677          2200 2
; 1678          2201 2      WHILE CLI$GET_VALUE( $DESCRIPTOR('INFILE'), DESC ) DO
; 1679          2202 2
; 1680          2203 2          IF COPY$FIND_INPUT_FILE( DESC )                     ! Parse the input file-specification.
; 1681          2204 2          THEN
; 1682          2205 2              REPORT_BYPASS( MSG$_NOTCOPIED );                ! Report that the file was not processed.
; 1683          2206 2
; 1684          2207 2    !
; 1685          2208 2    ! Return to the caller.
; 1686          2209 2    !
; 1687          2210 2
; 1688          2211 2      RETURN true;                                            ! Return to the caller.
; 1689          2212 2
; 1690          2213 1      END;


                                                            .PSECT  $PLIT$,NOWRT,NOEXE,2

                        45  4C  49  46  4E  49  0004C P.AAH:  .ASCII  \INFILE\                                    ;
                                                00052         .BLKB   2
                                     00000006   00054 P.AAG:  .LONG   6                                           ;
                                     00000000'  00058         .ADDRESS P.AAH                                      ;


                                                            .PSECT  $CODE$,NOWRT,2

                                         003C   00000 BYPASS_CONCAT:
                                                              .WORD   Save R2,R3,R4,R5                            ; 2120
                              5E          08 C2 00002         SUBL2   #8, SP                                      ;
   08            00           6E          00 2C 00005         MOVC5   #0, (SP), #0, #8, DESC                      ; 2166
                                          6E    0000A                                                             ;
                          03  AE          02 90 0000B         MOVB    #2, DESC+3                                  ; 2167
                          06       0000'  CF   E8 0000F         BLBS    COPY$CLI_STATUS, 1$                           : 2173
          49     0000'    CF              03   E1 00014         BBC     #3, COPY$SEM_STATUS+2, 5$                     : 2174
                 0000'    CF              08   8A 0001A 1$:     BICB2   #8, COPY$SEM_STATUS+2                         : 2178
          19     0000'    CF              05   E1 0001F         BBC     #5, COPY$SEM_STATUS+2, 3$                     : 2184
                 0000'    CF              20   8A 00025         BICB2   #32, COPY$SEM_STATUS+2                        : 2187
                                   0000G  CF   95 0002A         TSTB    INFILE_NAM_BLK+3                              : 2189
                                          0E   13 0002E         BEQL    3$                                            :
                                   0000G  CF   94 00030         CLRB    INFILE_NAM_BLK+3                              : 2192
                          7E       11C0   8F   3C 00034         MOVZWL  #4544, -(SP)                                  : 2193
                 0000V    CF              01   FB 00039 2$:     CALLS   #1, REPORT_BYPASS                             :
                                          5E   DD 0003E 3$:     PUSHL   SP                                            : 2201
                                   0000'  CF   9F 00040         PUSHAB  P_AAG                                         :
             00000000G    00              02   FB 00044         CALLS   #2, CLI$GET_VALUE                             :
                          11              50   E9 0004B         BLBC    R0, 4$                                        :
                                          5E   DD 0004E         PUSHL   SP                                            : 2203
                 0000V    CF              01   FB 00050         CALLS   #1, COPY$FIND_INPUT_FILE                      :
                          E6              50   E9 00055         BLBC    R0, 3$                                        :
                          7E       11B8   8F   3C 00058         MOVZWL  #4536, -(SP)                                  : 2205
                                          DA   11 0005D         BRB     2$                                            :
                          50              01   D0 0005F 4$:     MOVL    #1, R0                                        : 2211
                                               04 00062         RET                                                   :
                                          50   D4 00063 5$:     CLRL    R0                                            : 2213
                                               04 00065         RET                                                   :

; Routine Size:  102 bytes,    Routine Base:  $CODE$ + 0699
```

```
1 GLOBAL ROUTINE COPY$FIND_INPUT_FILE ( INFILE_DESC : REF $BBLOCK ) =
1
1 !++
1 ! FUNCTIONAL DESCRIPTION:
1 !
1 !       This routine calls RMS to parse an input file-specification.
1 !
1 ! FORMAL PARAMETERS:
1 !
1 !       None
1 !
1 ! IMPLICIT INPUTS:
1 !
1 !       INFILE_FAB - Input file FAB
1 !       INFILE_NAM_BLK - Input file name block
1 !
1 ! IMPLICIT OUTPUTS:
1 !
1 !       INFILE_FAB - FNA and FNS fields filled in.
1 !
1 ! COMPLETION CODES:
1 !
1 !       OK = Successful parse
1 !       ERROR = Error from RMS parse
1 !
1 ! SIDE EFFECTS:
1 !
1 !       None
1 !
1 !--
1
2     BEGIN
2
2     OWN
2         find_file_context : INITIAL(0);              ! Context parameter for LIB$FIND_FILE
2
2     LOCAL
2         resultant_name_desc : $BBLOCK[ DSC$C_S_BLN ],    ! Descriptor for filespec returned by LIB$FIND_FILE
2         find_file_nam : REF $BBLOCK[],                   ! Pointer to NAM block used by LIB$FIND_FILE
2         status;                                          ! Status returned by LIB$FIIND_FILE
2
2     BIND
2         find_file_fab = find_file_context : REF $BBLOCK[];
2
2
2
2     ! Initialize the descriptor for the resultant name string.
2     !
2     CH$FILL( 0, DSC$C_S_BLN, resultant_name_desc );
2     resultant_name_desc[ DSC$B_CLASS ] = DSC$K_CLASS_D;
2
2
2     ! Zero the expanded name sting length, so that COPY$INOPN_ERR can determine
2     ! if the expanded string was created by RMS or not.
2     !
2     INFILE_NAM_BLK[NAM$B_ESL] = 0;
2
```

```
; 1749       2271 2
; 1750       2272 2        ! Call LIB$FIND_FILE to locate the file.  If something other than success is
; 1751       2273 2        ! returned, then check to see if it is something we care about.  NMF, no
; 1752       2274 2        ! more files doesn't matter,  for any other error condition COPY should
; 1753       2275 2        ! issue a message.
; 1754       2276 2        !
; 1755       2277 3        IF NOT ( status = LIB$FIND_FILE( .infile_desc, resultant_name_desc,
; 1756       2278 3                    find_file_context, 0, 0, 0, %ref(2)))
; 1757       2279 3        THEN
; 1758       2280 3            BEGIN
; 1759       2281 3            IF .status NEQ RMS$_NMF
; 1760       2282 3            THEN
; 1761       2283 3                COPY$INOPN_ERR( .find_file_context );
; 1762       2284 3            RETURN .status;
; 1763       2285 2            END;
; 1764       2286 2
; 1765       2287 2
; 1766       2288 2        ! Copy the information from the resultant name string descripitor into
; 1767       2289 2        ! the FAB's file name and the NAM block's resultant name descriptor fields.
; 1768       2290 2        ! Also, copy the file name status bits into the input file's NAM block and
; 1769       2291 2        ! copy the FID of the found file into the input file's name block. (COPY
; 1770       2292 2        ! does an open by name block. This guarantees that the correct file is
; 1771       2293 2        ! opened.).  Then return to the caller.
; 1772       2294 2
; 1773       2295 2        infile_fab[ FAB$L_FNA ]      = .resultant_name_desc[ DSC$A_POINTER ];
; 1774       2296 2        infile_fab[ FAB$B_FNS ]      = .resultant_name_desc[ DSC$W_LENGTH ];
; 1775       2297 2        infile_nam_blk[ NAM$B_RSL ] = .resultant_name_desc[ DSC$W_LENGTH ];
; 1776       2298 2        in_name_desc[ 0 ]            = .infile_nam_blk[ NAM$B_RSL ];
; 1777       2299 2        CH$MOVE(.infile_fab[FAB$B_FNS], .infile_fab[FAB$L_FNA], .in_name_desc[1]);
; 1778       2300 2
; 1779       2301 2
; 1780       2302 2        find_file_nam = .find_file_fab[ FAB$L_NAM ];
; 1781       2303 2        infile_nam_blk[ NAM$L_FNB ]      = .find_file_nam[ NAM$L_FNB ];
; 1782       2304 2        infile_nam_blk[ NAM$W_FID_NUM ] = .find_file_nam[ NAM$W_FID_NUM ];
; 1783       2305 2        infile_nam_blk[ NAM$W_FID_SEQ ] = .find_file_nam[ NAM$W_FID_SEQ ];
; 1784       2306 2        infile_nam_blk[ NAM$W_FID_RVN ] = .find_file_nam[ NAM$W_FID_RVN ];
; 1785       2307 2        CH$MOVE( NAM$S_DVI, find_file_nam[NAM$T_DVI], infile_nam_blk[NAM$T_DVI] );
; 1786       2308 2
; 1787       2309 2        RETURN ok;
; 1788       2310 2
; 1789       2311 1        END;


                                                        .PSECT  $OWN$,NOEXE,2

                                    00000000  00000 FIND_FILE_CONTEXT:
                                                        .LONG   0                                                ;

                                                FIND_FILE_FAB=      FIND_FILE_CONTEXT


                                                        .PSECT  $CODE$,NOWRT,2

                                         00FC 00000     .ENTRY  COPY$FIND_INPUT_FILE, Save R2,R3,R4,R5,R6,- ; 2214
                                                                R7                                               ;
                        57     0000'  CF   9E 00002     MOVAB   FIND_FILE_CONTEXT, R7                            ;
```

```
                                   56       0000G  CF  9E 00007          MOVAB   INFILE_NAM_BLK+3, R6                          ;
                                   5E              0C  C2 0000C          SUBL2   #12, SP                                       ;
        08             00          6E              00  2C 0000F          MOVC5   #0, (SP), #0, #8, RESULTANT_NAME_DESC         ; 2262
                                            04     AE     00014                                                                ;
                              07   AE              02  90 00016          MOVB    #2, RESULTANT_NAME_DESC+3                     ; 2263
                                            08     A6  94 0001A          CLRB    INFILE_NAM_BLK+11                             ; 2269
                                   6E              02  D0 0001D          MOVL    #2, (SP)                                      ; 2278
                                                   5E  DD 00020          PUSHL   SP                                            ;
                                                   7E  7C 00022          CLRQ    -(SP)                                         ; 2277
                                                   7E  D4 00024          CLRL    -(SP)                                         ;
                                                   57  DD 00026          PUSHL   R7                                            ;
                                            18     AE  9F 00028          PUSHAB  RESULTANT_NAME_DESC                           ;
                                            04     AC  DD 0002B          PUSHL   INFILE_DESC                                   ;
                        000000000G 00              07  FB 0002E          CALLS   #7, LIB$FIND_FILE                             ;
                                   52              50  D0 00035          MOVL    R0, STATUS                                    ;
                                   14              52  E8 00038          BLBS    STATUS, 2$                                    ;
                        000182CA   8F              52  D1 0003B          CMPL    STATUS, #99018                                ; 2281
                                                   07  13 00042          BEQL    1$                                            ;
                                                   67  DD 00044          PUSHL   FIND_FILE_CONTEXT                             ; 2283
                            0000V  CF              01  FB 00046          CALLS   #1, COPY$INOPN_ERR                            ;
                                   50              52  D0 0004B  1$:     MOVL    STATUS, R0                                    ; 2284
                                                   04     0004E          RET                                                   ;
                            0000G  CF       08     AE  D0 0004F  2$:     MOVL    RESULTANT_NAME_DESC+4, INFILE_FAB+44          ; 2295
                            0000G  CF       04     AE  90 00055          MOVB    RESULTANT_NAME_DESC, INFILE_FAB+52            ; 2296
                                   66       04     AE  90 0005B          MOVB    RESULTANT_NAME_DESC, INFILE_NAM_BLK+3         ; 2297
                            0000G  CF              66  9A 0005F          MOVZBL  INFILE_NAM_BLK+3, IN_NAME_DESC                ; 2298
                                   50       0000G  CF  9A 00064          MOVZBL  INFILE_FAB+52, R0                             ; 2299
                  0000G  DF  0000G DF              50  28 00069          MOVC3   R0, @INFILE_FAB+44, @IN_NAME_DESC+4           ;
                                   50              67  D0 00071          MOVL    FIND_FILE_FAB, R0                             ; 2302
                                   50       28     A0  D0 00074          MOVL    40(R0), FIND_FILE_NAM                         ;
                              31   A6       34     A0  D0 00078          MOVL    52(FIND_FILE_NAM), INFILE_NAM_BLK+52          ; 2303
                              21   A6       24     A0  D0 0007D          MOVL    36(FIND_FILE_NAM), INFILE_NAM_BLK+36          ; 2304
                              25   A6       28     A0  B0 00082          MOVW    40(FIND_FILE_NAM), INFILE_NAM_BLK+40          ; 2306
                 11    A6     14   A0              10  28 00087          MOVC3   #16, 20(FIND_FILE_NAM), INFILE_NAM_BLK+20     ; 2307
                                   50              01  D0 0008D          MOVL    #1, R0                                        ; 2309
                                                   04     00090          RET                                                   ; 2311

; Routine Size:  145 bytes,    Routine Base:  $CODE$ + 06FF
```

```
1 GLOBAL ROUTINE COPY$CALC_ALQ =                                   ! Allocation quantity calculation routine
1
1 !++
1 ! FUNCTIONAL DESCRIPTION:
1 !
1 !       This routine determines the output file allocation/extension quantity.
1 !
1 ! FORMAL PARAMETERS:
1 !
1 !       None
1 !
1 ! IMPLICIT INPUTS:
1 !
1 !       EXTEND_OUTFILE  - Output file extension indicator
1 !       INFILE_FAB      - Input file FAB
1 !       INFILE_XABALL   - Input file allocation XAB
1 !       INFILE_XABFHC   - Input file header characteristics XAB
1 !       COPY$CLI_STATUS bit TRUNCATE_BIT
1 !               means /TRUNCATE was specified
1 !       ALLOC_VALUE     - contains a value if /ALLOCATION was specified.
1 !       COPY_TRUN_QUAL  - CLI data block for the truncate qualifier; the
1 !                         "explicit bit" will be set if /NOTRUNCATE was
1 !                         specified on the input line
1 !
1 ! IMPLICIT OUTPUTS:
1 !
1 !       None
1 !
1 ! ROUTINE VALUE:
1 !
1 !       Size of the input file (i.e., number of blocks)
1 !
1 ! SIDE EFFECTS:
1 !
1 !       None
1 !
1 !--
1
2     BEGIN
2
2     LOCAL
2         ALQ;                                                    ! Temporary allocation quantity
2 !
2 ! Return a zero allocation size if the output file is not a disk and it is being extended.
2 !
2
2     IF .EXTEND_OUTFILE AND                                      ! If the output file is being extended
3         (NOT .OUTFILE_FAB[$FAB_DEV(FOD)] OR                     ! and it is not a file structured device
3          .OUTFILE_FAB[$FAB_DEV(SQD)])                           ! or it is a magnetic tape,
2     THEN                                                        !
2         RETURN 0;                                               ! return a zero allocation size to the caller.
2
2 !
2 ! Determine the output file allocation size from the size and organization of the input file.
2 !
2
```

```
    IF NOT .INFILE_FAB[$FAB_DEV(FOD)] OR                    ! If the input device is not file structured
        .INFILE_FAB[$FAB_DEV(SQD)]                          ! or if it is a magnetic tape,
    THEN                                                    !
        ALQ = DEFAULT_ALLOC                                 ! assume a default input file size.
    ELSE
        BEGIN
!
!       If the input file is a non-contiguous sequential file and /NOTRUNCATE was not explicitly given
!               or
!       the the input file is being appened to an existing file,
!               or
!       if /TRUNCATE and no /ALLOCATION was given,
!       the file should be truncated.  Otherwise, use the allocation of the input file as the size of
!       the output file.
!
        IF  (
            (
            .INFILE_FAB[ FAB$B_ORG ] EQL FAB$C_SEQ
                AND NOT
            ( .TRUNCATE_NEGATED OR .NEG_TRUNCATE_QUAL )
            )
                AND
            ( NOT .INFILE_XABALL[ XAB$V_CTG ] OR .EXTEND_OUTFILE )
            )
                OR
            ( (.TRUNCATE_QUAL OR .LOC_TRUNCATE_QUAL) AND .CURR_ALLOCATION_VALUE EQL 0 )
        THEN                                                !
            IF .INFILE_XABFHC[XAB$W_FFB] EQL 0              ! calculate only enough space to hold the actual
            THEN                                            ! data in the input file.  Note that this calculatio
                ALQ = .INFILE_XABFHC[XAB$L_EBK] - 1         ! includes the final block only if it actually
            ELSE                                            ! contains some data.
                ALQ = .INFILE_XABFHC[XAB$L_EBK]             !
        ELSE
            ALQ = .INFILE_XABFHC[XAB$L_HBK];                ! Otherwise, pickup the actual size of the input fil
        END;

    IF .EXTEND_OUTFILE                                      ! If the output file is being extended,
    THEN                                                    !
        ALQ = .OUTFILE_XABFHC[XAB$L_EBK] + .ALQ -           ! subtract the remaining output file space
                .OUTFILE_XABFHC[XAB$L_HBK];                 ! from the calculated extension quantity.

!
! Return the calculated allocation (or extension) quantity to the caller.
!

    IF .ALQ GEQ 0                                           ! If the calculated allocation/extension quantity
    THEN                                                    ! is greater than or equal to zero,
        RETURN .ALQ                                         ! return that value to the caller.
    ELSE
        RETURN 0;                                           ! Otherwise, return a zero value to the caller.

    END;
```

```
                                                      0004 00000          .ENTRY  COPY$CALC_ALQ, Save R2                        ; 2312
                               52        0000G   CF   9E 00002            MOVAB   INFILE_XABFHC+16, R2                          ;
         51      0000'   CF    01                07   EF 00007            EXTZV   #7, #1, COPY$SEM_STATUS+2, R1                 ; 2359
                               0C                51   E9 0000E            BLBC    R1, 1$                                        ;
                     67  0000G   CF              06   E1 00011            BBC     #6, OUTFILE_FAB+65, 11$                       ; 2360
                     61  0000G   CF              05   E0 00017            BBS     #5, OUTFILE_FAB+64, 11$                       ; 2361
                     06  0000G   CF              06   E1 0001D 1$:        BBC     #6, INFILE_FAB+65, 2$                         ; 2369
                     04  0000G   CF              05   E1 00023            BBC     #5, INFILE_FAB+64, 3$                         ; 2370
                                                 50   D4 00029 2$:        CLRL    ALQ                                           ; 2372
                                                 40   11 0002B            BRB     9$                                            ;
                                         0000G   CF   95 0002D 3$:        TSTB    INFILE_FAB+29                                 ; 2386
                                                 14   12 00031            BNEQ    4$                                            ;
                     0E  0000'   CF              06   E0 00033            BBS     #6, COPY$CLI_STATUS+5, 4$                     ; 2388
                               09        0000'   CF   E8 00039            BLBS    COPY$CLI_STATUS+6, 4$                         ;
                                         0000G   CF   95 0003E            TSTB    INFILE_XABALL+8                               ; 2391
                                                 15   18 00042            BGEQ    6$                                            ;
                               12                51   E8 00044            BLBS    R1, 6$                                        ;
                     06  0000'   CF              05   E0 00047 4$:        BBS     #5, COPY$CLI_STATUS+5, 5$                     ; 2394
                                         0000'   CF   95 0004D            TSTB    COPY$CLI_STATUS+5                             ;
                                                 16   18 00051            BGEQ    8$                                            ;
                                         0000G   CF   D5 00053 5$:        TSTL    CURR_ALLOCATION_VALUE                         ;
                                                 10   12 00057            BNEQ    8$                                            ;
                                            04   A2   B5 00059 6$:        TSTW    INFILE_XABFHC+20                              ; 2396
                                                 06   12 0005C            BNEQ    7$                                            ;
                     50        62                01   C3 0005E            SUBL3   #1, INFILE_XABFHC+16, ALQ                     ; 2398
                                                 09   11 00062            BRB     9$                                            ;
                               50                62   D0 00064 7$:        MOVL    INFILE_XABFHC+16, ALQ                         ; 2400
                                                 04   11 00067            BRB     9$                                            ; 2396
                               50           FC   A2   D0 00069 8$:        MOVL    INFILE_XABFHC+12, ALQ                         ; 2402
                               0C                51   E9 0006D 9$:        BLBC    R1, 10$                                       ; 2405
                     51        50        0000G   CF   C1 00070            ADDL3   OUTFILE_XABFHC+16, ALQ, R1                    ; 2407
                     50        51        0000G   CF   C3 00076            SUBL3   OUTFILE_XABFHC+12, R1, ALQ                    ; 2408
                                                 02   18 0007C 10$:       BGEQ    12$                                           ; 2414
                                                 50   D4 0007E 11$:       CLRL    R0                                            ; 2420
                                                      04 00080 12$:       RET                                                   ;

; Routine Size:  129 bytes,    Routine Base:  $CODE$ + 0790
```

```
1 ROUTINE REPORT_NAMES                                           ! Report the results of a file copy
1                 : NOVALUE =
1
1 !++
1 ! FUNCTIONAL DESCRIPTION:
1 !
1 !       This routine reports the results of copying a single input file
1 !       to the output file.
1 !
1 ! FORMAL PARAMETERS:
1 !
1 !       None
1 !
1 ! IMPLICIT INPUTS:
1 !
1 !       LOG - Indicator tested to see if activity reporting desired
1 !       EXTEND_OUTFILE - Indicator tested to see if input concatenation is active.
1 !       IN_NAME_DESC - Input file name descriptor
1 !       OUT_NAME_DESC - Output file name descriptor
1 !       BLOCK_COUNT - Number of input file blocks copied
1 !       RECORD_COUNT - Number of input file records copied
1 !       INFILE_FAB - Address of input file FAB
1 !
1 ! IMPLICIT OUTPUTS:
1 !
1 !       None
1 !
1 ! ROUTINE VALUE:
1 !
1 !       None
1 !
1 ! SIDE EFFECTS:
1 !
1 !       None
1 !
1 !--
1
2    BEGIN
2
2    LOCAL
2       ptr,                                                     ! Temporary variables for character searching
2       address,
2       size;
2
2 !
2 ! Determine which message, if any, is needed.
2 !
2
2    IF NOT .LOG_MSG_QUAL                                        ! If activity reporting is not requested,
2    THEN                                                        !
2       RETURN;                                                  ! return to the caller.
2
2 !
2 ! If this is a record oriented device (not network), the messages should
2 ! include only the device name.
2 !
```

```
  2
  2        IF .infile_fab [$FAB_DEV(rec)]
  2            AND NOT .infile_fab [$FAB_DEV(net)]
  2        THEN
  3            BEGIN
  3            size = .in_name_desc[0];
  3            address = .in_name_desc[1];
  3            ptr = CH$FIND_CH(.size,.address,':');
  3            IF .ptr NEQ 0           ! If there is anything past the device, remove it
  3            THEN
  3                in_name_desc[0] =  .ptr - .address + 1;
  2            END;
  2
  2
  2        IF NOT .EXTEND_OUTFILE                                  ! Test the record mode indicator to see
  2                                                                ! if this is the primary input file or a
  2                                                                ! concatenated input file.
  2 !
  2 ! Create a "copied" message if the input file just copied was
  2 ! the first file copied into the output file.
  2 !
  2
  2        THEN                                                    !
  2            IF .BLOCK_COUNT NEQ 0                               ! If the input file was copied in block mode,
  2            THEN                                                !
P 2                PUT_MESSAGE( MSG$_COPIEDB,                      ! signal "file copied" with the following arguments:
P 2                        3,                                      !    Number of message arguments
P 2                        IN_NAME_DESC,                           !    Address of input file name descriptor
P 2                        OUT_NAME_DESC,                          !    Address of output file name descriptor
  2                        .BLOCK_COUNT )                          !    Number of blocks copied
  2
  2            ELSE                                                ! Otherwise,
  3                IF (.RECORD_COUNT NEQ 0) OR NOT (LIB$CHECK_DIR (INFILE_FAB)) ! If the input file is not 0 record
  3                                                                    !   is not a directory file
  2                THEN
P 2                        PUT_MESSAGE( MSG$_COPIEDR,              ! signal "file copied" with the following arguments:
P 2                                3,                              !    Number of message arguments
P 2                                IN_NAME_DESC,                   !    Address of input file name descriptor
P 2                                OUT_NAME_DESC,                  !    Address of output file name descriptor
  2                                .RECORD_COUNT )                 !    Number of records copied
  2
  2                ELSE                                            ! Otherwise, its a directory file
P 2                        PUT_MESSAGE( MSG$_CREATED,              ! signal "created" with the following arguments:
P 2                                1,                              !    number of message arguments
  2                                OUT_NAME_DESC )                 !    address of output file descriptor
  2 !
  2 ! Create an "appended" message if the input file just copied was
  2 ! appended to an existing output file.
  2 !
  2
  2        ELSE                                                    !
  2            IF .BLOCK_COUNT NEQ 0                               ! If the input file was copied in block mode,
  2            THEN                                                !
P 2                PUT_MESSAGE( MSG$_APPENDEDB,                    ! signal "file appended" with the following argument
P 2                        3,                                      !    Number of message arguments
P 2                        IN_NAME_DESC,                           !    Address of input file name descriptor
P 2                        OUT_NAME_DESC,                          !    Address of output file name descriptor
; 2015            2535  2                  .BLOCK_COUNT )                       !    Number of blocks copied
; 2016            2536  2
; 2017            2537  2        ELSE                                           ! Otherwise,
; 2018          P 2538  2            PUT_MESSAGE( MSG$_APPENDEDR,               ! signal "file appended" with the following argument
; 2019          P 2539  2                    3,                                 !    Number of message arguments
; 2020          P 2540  2                    IN_NAME_DESC,                      !    Address of input file name descriptor
; 2021          P 2541  2                    OUT_NAME_DESC,                     !    Address of output file name descriptor
; 2022            2542  2                    .RECORD_COUNT );                   !    Number of records copied
; 2023            2543  2
; 2024            2544  2 !
; 2025            2545  2 ! Return to the caller.
; 2026            2546  2 !
; 2027            2547  2
; 2028            2548  2   RETURN;                                             ! Return to the caller.
; 2029            2549  2
; 2030            2550  1   END;



                                        007C 00000 REPORT_NAMES:
                                                             .WORD   Save R2,R3,R4,R5,R6                        ; 2421
                   56 00000000G  00   9E 00002               MOVAB   LIB$SIGNAL, R6                             ;
                   55     0000G  CF   9E 00009               MOVAB   OUT_NAME_DESC, R5                          ;
                   54     0000'  CF   9E 0000E               MOVAB   RECORD_COUNT, R4                           ;
                   53     0000G  CF   9E 00013               MOVAB   IN_NAME_DESC, R3                           ;
     01       14   A4             01  E0 00018               BBS     #1, COPY$CLI_STATUS, 1$                    ; 2469
                                  04 0001D                   RET                                                ;
                   20     0000G  CF   E9 0001E 1$:           BLBC    INFILE_FAB+64, 3$                          ; 2479
     1A    0000G   CF             05  E0 00023               BBS     #5, INFILE_FAB+65, 3$                      ; 2480
                   50             63  D0 00029               MOVL    IN_NAME_DESC, SIZE                         ; 2483
                   52        04   A3  D0 0002C               MOVL    IN_NAME_DESC+4, ADDRESS                    ; 2484
     62            50             3A  3A 00030               LOCC    #58, SIZE, (ADDRESS)                       ; 2485
                                  02  12 00034               BNEQ    2$                                         ;
                                  51  D4 00036               CLRL    R1                                         ;
                                  51  D5 00038 2$:           TSTL    PTR                                        ; 2486
                                  07  13 0003A               BEQL    3$                                         ;
                   51             52  C2 0003C               SUBL2   ADDRESS, R1                                ; 2488
                   63        01   A1  9E 0003F               MOVAB   1(R1), IN_NAME_DESC                        ;
                   50        FC   A4  D0 00043 3$:           MOVL    BLOCK_COUNT, R0                            ; 2501
                             32   A4  95 00047               TSTB    COPY$SEM_STATUS+2                          ; 2492
                                  44  19 0004A               BLSS    7$                                         ;
                                  50  D5 0004C               TSTL    R0                                         ; 2501
                                  0D  13 0004E               BEQL    4$                                         ;
                                  50  DD 00050               PUSHL   R0                                         ; 2507
                                  28  BB 00052               PUSHR   #^M<R3,R5>                                 ;
                                  03  DD 00054               PUSHL   #3                                         ;
                   7E      1061   8F  3C 00056               MOVZWL  #4193, -(SP)                               ;
                                  4F  11 0005B               BRB     9$                                         ;
                                  64  D5 0005D 4$:           TSTL    RECORD_COUNT                               ; 2510
                                  0E  12 0005F               BNEQ    5$                                         ;
                          0000G   CF  9F 00061               PUSHAB  INFILE_FAB                                 ;
         00000000G 00             01  FB 00065               CALLS   #1, LIB$CHECK_DIR                          ;
                   0D             50  E8 0006C               BLBS    R0, 6$                                     ;
                                  64  DD 0006F 5$:           PUSHL   RECORD_COUNT                               ; 2517
                                  28  BB 00071               PUSHR   #^M<R3,R5>                                 ;
```

```
                               03   DD 00073          PUSHL    #3                                      ;
              7E       1069    8F   3C 00075          MOVZWL   #4201, -(SP)                            ;
                               30   11 0007A          BRB      9$                                      ;
                               55   DD 0007C 6$:      PUSHL    R5                                      ; 2522
                               01   DD 0007E          PUSHL    #1                                      ;
              7E       1073    8F   3C 00080          MOVZWL   #4211, -(SP)                            ;
       0000V  CF               01   FB 00085          CALLS    #1, COPY$MSG_NUMBER                     ;
                               50   DD 0008A          PUSHL    R0                                      ;
              66               03   FB 0008C          CALLS    #3, LIB$SIGNAL                          ;
                                    04 0008F          RET                                              ; 2501
                               50   D5 00090 7$:      TSTL     R0                                      ; 2529
                               0D   13 00092          BEQL     8$                                      ;
                               50   DD 00094          PUSHL    R0                                      ; 2535
                               28   BB 00096          PUSHR    #^M<R3,R5>                              ;
                               03   DD 00098          PUSHL    #3                                      ;
              7E       1001    8F   3C 0009A          MOVZWL   #4097, -(SP)                            ;
                               0B   11 0009F          BRB      9$                                      ;
                               64   DD 000A1 8$:      PUSHL    RECORD_COUNT                            ; 2542
                               28   BB 000A3          PUSHR    #^M<R3,R5>                              ;
                               03   DD 000A5          PUSHL    #3                                      ;
              7E       1009    8F   3C 000A7          MOVZWL   #4105, -(SP)                            ;
       0000V  CF               01   FB 000AC 9$:      CALLS    #1, COPY$MSG_NUMBER                     ;
                               50   DD 000B1          PUSHL    R0                                      ;
              66               05   FB 000B3          CALLS    #5, LIB$SIGNAL                          ;
                                    04 000B6          RET                                              ; 2550

; Routine Size:  183 bytes,    Routine Base:  $CODE$ + 0811
```

```
1 ROUTINE REPORT_BYPASS (                                         ! Report the bypassing of an input file
1                         NUMBER )                                ! Error number
1                 : NOVALUE =
1
1 !++
1 ! FUNCTIONAL DESCRIPTION:
1 !
1 !       This routine reports the name of an input file which has been bypassed.
1 !
1 ! FORMAL PARAMETERS:
1 !
1 !       NUMBER.rlu.v - Error number
1 !
1 ! IMPLICIT INPUTS:
1 !
1 !       INFILE_NAM_BLK - Input file name block
1 !       INFILE_NAME - Input file resultant name
1 !       INFILE_XNAME - Input file expanded name
1 !
1 ! IMPLICIT OUTPUTS:
1 !
1 !       None
1 !
1 ! ROUTINE VALUE:
1 !
1 !       None
1 !
1 ! SIDE EFFECTS:
1 !
1 !       None
1 !
1 !--
1
2     BEGIN
2
2     LOCAL
2         NAME_DESC :  VECTOR[2];                                 ! Input file name descriptor
2
2 !
2 ! Setup the input file name descriptor.
2 !
2
2     IF .INFILE_NAM_BLK[NAM$B_RSL] NEQ 0                         ! If RMS has setup a resultant name string,
2     THEN                                                        !
3         BEGIN                                                   !
3         NAME_DESC[0] = .INFILE_NAM_BLK[NAM$B_RSL];              ! setup the name descriptor to use
3         NAME_DESC[1] = INFILE_NAME;                             ! the resultant name string.
3         END
2     ELSE
3         BEGIN                                                   !
3         NAME_DESC[0] = .INFILE_NAM_BLK[NAM$B_ESL];              ! Otherwise, use the expanded name string.
3         NAME_DESC[1] = INFILE_XNAME;                            !
3         END;
2
2 !
2 ! Report the name of the input file which is being bypassed.
2 !
```

```
; 2089          2608  2
; 2090          2609  2     PUT_MESSAGEX( .NUMBER, 1, NAME_DESC );              ! Report the name of the input file.
; 2091          2610  2
; 2092          2611  2 !
; 2093          2612  2 ! Return to the caller.
; 2094          2613  2 !
; 2095          2614  2
; 2096          2615  2     RETURN;                                             ! Return to the caller.
; 2097          2616  2
; 2098          2617  1     END;



                                                   0004 00000 REPORT_BYPASS:
                                                                      .WORD    Save R2                                      ; 2551
                         52        0000V   CF   9E 00002              MOVAB    COPY$MSG_NUMBER, R2                          ;
                         5E           08   C2 00007                   SUBL2    #8, SP                                       ;
                         50        0000G   CF   9A 0000A              MOVZBL   INFILE_NAM_BLK+3, R0                         ; 2593
                                      0B   13 0000F                   BEQL     1$                                           ;
                         6E           50   D0 00011                   MOVL     R0, NAME_DESC                                ; 2596
                    04   AE        0000G   CF   9E 00014              MOVAB    INFILE_NAME, NAME_DESC+4                     ; 2597
                                      0B   11 0001A                   BRB      2$                                           ; 2593
                         6E        0000G   CF   9A 0001C 1$:          MOVZBL   INFILE_NAM_BLK+11, NAME_DESC                 ; 2601
                    04   AE        0000G   CF   9E 00021              MOVAB    INFILE_XNAME, NAME_DESC+4                    ; 2602
                                   04   AC   DD 00027 2$:             PUSHL    NUMBER                                       ; 2609
                         62           01   FB 0002A                   CALLS    #1, COPY$MSG_NUMBER                          ;
       7E         00     50           01   7A 0002D                   EMUL     #1, R0, #0, -(SP)                            ;
       50         50     8E           08   7B 00032                   EDIV     #8, (SP)+, R0, R0                            ;
                         04           50   D1 00037                   CMPL     R0, #4                                       ;
                                      14   13 0003A                   BEQL     3$                                           ;
                                      5E   DD 0003C                   PUSHL    SP                                           ;
                                      01   DD 0003E                   PUSHL    #1                                           ;
                                   04   AC   DD 00040                 PUSHL    NUMBER                                       ;
                         62           01   FB 00043                   CALLS    #1, COPY$MSG_NUMBER                          ;
                                      50   DD 00046                   PUSHL    R0                                           ;
                00000000G  00         03   FB 00048                   CALLS    #3, LIB$SIGNAL                               ;
                                           04 0004F                   RET                                                   ;
                                      5E   DD 00050 3$:               PUSHL    SP                                           ;
                                      01   DD 00052                   PUSHL    #1                                           ;
                                   04   AC   DD 00054                 PUSHL    NUMBER                                       ;
                         62           01   FB 00057                   CALLS    #1, COPY$MSG_NUMBER                          ;
                                      50   DD 0005A                   PUSHL    R0                                           ;
                00000000G  00         03   FB 0005C                   CALLS    #3, LIB$STOP                                 ;
                                           04 00063                   RET                                                   ; 2617

; Routine Size:  100 bytes,    Routine Base:  $CODE$ + 08C8
```

```
GLOBAL ROUTINE COPY$LOG_MSG (                                   ! Signal a COPY message
                  NUMBER )                                      ! Error number
                    : NOVALUE =

!++
! FUNCTIONAL DESCRIPTION:
!
!       This routine sends an informational message to the user if
!       activity reporting has been requested.
!
! FORMAL PARAMETERS:
!
!       NUMBER.rlu.v - error number
!
! IMPLICIT INPUTS:
!
!       LOG_MSG - Activity reporting indicator
!       OUTFILE_COUNT - Number of output files created
!       OUT_NAME_DESC - Output file name descriptor
!
! IMPLICIT OUTPUTS:
!
!       None
!
! ROUTINE VALUE:
!
!       None
!
! SIDE EFFECTS:
!
!       None
!
!--

    BEGIN

!
! Return to the caller if activity reporting has not been requested.
!

    IF NOT .LOG_MSG_QUAL                                        ! If activity reporting is not requested,
    THEN                                                        !
        RETURN;                                                 ! return to the caller.

!
! Call FAO to format the error message in the message buffer.
!

    SELECTONE .NUMBER OF                                        ! Select error message processing based
        SET                                                     ! on the actual error number.

        [MSG$_NEWFILES]:
            IF .OUTFILE_COUNT GEQU 2                            ! If at least 2 files was created,
            THEN                                                !
                PUT_MESSAGE( MSG$_NEWFILES,                     ! signal "<number> files created" with the following
                        1,                                      !    number of message arguments
                        .OUTFILE_COUNT );                       !    number of output files created
```

```
; 2157          2675  2              [MSG$_REPLACED, MSG$_OVERLAY, MSG$_CREATED]:
; 2158          2676  2                  PUT_MESSAGEX( .NUMBER,                           ! signal the message with the following arguments:
; 2159        P 2677  2                                1,                                 !    number of message arguments
; 2160        P 2678  2                                OUT_NAME_DESC );                   !    address of the output name descriptor
; 2161          2679  2
; 2162          2680  2              [OTHERWISE]:
; 2163          2681  2                  PUT_MESSAGEX( .NUMBER );                         ! Signal the appropriate message.
; 2164          2682  2              TES;
; 2165          2683  2
; 2166          2684  2      !
; 2167          2685  2      ! Return to the caller.
; 2168          2686  2      !
; 2169          2687  2
; 2170          2688  2          RETURN;                                                  ! Return to the caller.
; 2171          2689  2
; 2172          2690  2
; 2173          2691  1          END;



                                                   003C 00000        .ENTRY  COPY$LOG_MSG, Save R2,R3,R4,R5                 ; 2618
                     55 00000000G           00  9E 00002        MOVAB   LIB$STOP, R5                                           ;
                     54 00000000G           00  9E 00009        MOVAB   LIB$SIGNAL, R4                                         ;
                     53     0000V           CF  9E 00010        MOVAB   COPY$MSG_NUMBER, R3                                    ;
          01   0000'  CF                    01  E0 00015        BBS     #1, COPY$CLI_STATUS, 1$                                ; 2658
                                                04 0001B        RET                                                            ;
                     52        04           AC  D0 0001C  1$:   MOVL    NUMBER, R2                                             ; 2666
             00001091 8F                    52  D1 00020        CMPL    R2, #4241                                              ; 2669
                                            15  12 00027        BNEQ    3$                                                     ;
                     02     0000'           CF  D1 00029        CMPL    OUTFILE_COUNT, #2                                      ; 2670
                                            01  1E 0002E        BGEQU   2$                                                     ;
                                                04 00030        RET                                                            ;
                            0000'           CF  DD 00031  2$:   PUSHL   OUTFILE_COUNT                                          ; 2674
                                            01  DD 00035        PUSHL   #1                                                     ;
                     7E     1091            8F  3C 00037        MOVZWL  #4241, -(SP)                                           ;
                                            37  11 0003C        BRB     5$                                                     ;
             00001073 8F                    52  D1 0003E  3$:   CMPL    R2, #4211                                              ; 2676
                                            12  13 00045        BEQL    4$                                                     ;
             000010AB 8F                    52  D1 00047        CMPL    R2, #4267                                              ;
                                            09  13 0004E        BEQL    4$                                                     ;
             000010BB 8F                    52  D1 00050        CMPL    R2, #4283                                              ;
                                            36  12 00057        BNEQ    7$                                                     ;
                                            52  DD 00059  4$:   PUSHL   R2                                                     ; 2679
                     63                     01  FB 0005B        CALLS   #1, COPY$MSG_NUMBER                                    ;
     7E        00    50                     01  7A 0005E        EMUL    #1, R0, #0, -(SP)                                      ;
     50        50    8E                     08  7B 00063        EDIV    #8, (SP)+, R0, R0                                      ;
                     04                     50  D1 00068        CMPL    R0, #4                                                 ;
                                            11  13 0006B        BEQL    6$                                                     ;
                            0000G           CF  9F 0006D        PUSHAB  OUT_NAME_DESC                                          ;
                                            01  DD 00071        PUSHL   #1                                                     ;
                                            52  DD 00073        PUSHL   R2                                                     ;
                     63                     01  FB 00075  5$:   CALLS   #1, COPY$MSG_NUMBER                                    ;
                                            50  DD 00078        PUSHL   R0                                                     ;
                     64                     03  FB 0007A        CALLS   #3, LIB$SIGNAL                                         ;
                                                04 0007D        RET                                                            ;
                                                0000G   CF   9F 0007E 6$:      PUSHAB  OUT_NAME_DESC                                          ;
                                                        01   DD 00082          PUSHL   #1                                                     ;
                                                        52   DD 00084          PUSHL   R2                                                     ;
                                          63            01   FB 00086          CALLS   #1, COPY$MSG_NUMBER                                    ;
                                                        50   DD 00089          PUSHL   R0                                                     ;
                                          65            03   FB 0008B          CALLS   #3, LIB$STOP                                           ;
                                                             04 0008E          RET                                                            ;
                                                        52   DD 0008F 7$:      PUSHL   R2                                                     ; 2682
                                          63            01   FB 00091          CALLS   #1, COPY$MSG_NUMBER                                    ;
            7E            00              50            01   7A 00094          EMUL    #1, R0, #0, -(SP)                                      ;
            50            50              8E            08   7B 00099          EDIV    #8, (SP)+, R0, R0                                      ;
                                          04            50   D1 0009E          CMPL    R0, #4                                                 ;
                                                        08   13 000A1          BEQL    8$                                                     ;
                                                        52   DD 000A3          PUSHL   R2                                                     ;
                                          63            01   FB 000A5          CALLS   #1, COPY$MSG_NUMBER                                    ;
                                                        50   DD 000A8          PUSHL   R0                                                     ;
                                          64            01   FB 000AA          CALLS   #1, LIB$SIGNAL                                         ;
                                                             04 000AD          RET                                                            ;
                                                        52   DD 000AE 8$:      PUSHL   R2                                                     ;
                                          63            01   FB 000B0          CALLS   #1, COPY$MSG_NUMBER                                    ;
                                                        50   DD 000B3          PUSHL   R0                                                     ;
                                          65            01   FB 000B5          CALLS   #1, LIB$STOP                                           ;
                                                             04 000B8          RET                                                            ; 2691

; Routine Size:  185 bytes,    Routine Base:  $CODE$ + 092C
```

```
1 GLOBAL ROUTINE COPY$INOPN_ERR (                                  ! RMS input open error action routine
1                 FAB_RAB_ADDRESS )                                ! Address of associated FAB or RAB
1                 : NOVALUE =
1
1 !++
1 ! FUNCTIONAL DESCRIPTION:
1 !
1 !       This RMS error action routine sends an input open error message to the user.
1 !
1 ! FORMAL PARAMETERS:
1 !
1 !       FAB_RAB_ADDRESS.ra.v - Address of the associated FAB or RAB
1 !
1 ! IMPLICIT INPUTS:
1 !
1 !       Input file name block
1 !       Input file name after open
1 !       Input file name before open
1 !       Input file cli descriptor
1 !
1 ! IMPLICIT OUTPUTS:
1 !
1 !       None
1 !
1 ! ROUTINE VALUE:
1 !
1 !       None
1 !
1 ! SIDE EFFECTS:
1 !
1 !       None
1 !
1 !--
1
2     BEGIN
2
2     BIND
2         FAB_RAB = .FAB_RAB_ADDRESS :  BLOCK[,BYTE];      ! Redefine routine parameter.
2
2     LOCAL
2         MESSAGE_ID,                                      ! Local message identifier
2         NAM_BLK : REF $BBLOCK[],                         ! Pointer to NAM block
2         NAME_DESC :  VECTOR[2];                          ! Input file name descriptor
2
2 !
2 ! Fillin the file name descriptor with the most complete name possible.
2 !
2
2     NAM_BLK = .FAB_RAB[FAB$L_NAM];
2
2     IF .NAM_BLK[NAM$B_RSL] NEQ 0                         ! If a resultant name string exists,
2     THEN                                                 !
3         BEGIN                                            !
3         MESSAGE_ID = MSG$_OPENIN;                        ! indicate an open error
3         NAME_DESC[0] = .NAM_BLK[NAM$B_RSL];              ! and fillin the resultant name length
3         NAME_DESC[1] = .NAM_BLK[NAM$L_RSA];              ! and address.
3         END
```

```
; 2232          2749  2     ELSE
; 2233          2750  2         IF .NAM_BLK[NAM$B_ESL] NEQ 0                       ! If RMS created an expanded string
; 2234          2751  2         THEN                                               ! but coundn't open the file,
; 2235          2752  3             BEGIN
; 2236          2753  3             MESSAGE_ID = MSG$_OPENIN;                      ! indicate an open error
; 2237          2754  3             NAME_DESC[0] = .NAM_BLK[NAM$B_ESL];            ! and fillin the expanded name length
; 2238          2755  3             NAME_DESC[1] = .NAM_BLK[NAM$L_ESA];            ! and address.
; 2239          2756  3             END
; 2240          2757  2         ELSE
; 2241          2758  3             BEGIN
; 2242          2759  3             MESSAGE_ID = MSG$_OPENINX;                     ! Otherwise, indicate a fatal open error
; 2243          2760  3             NAME_DESC[0] = .INFILE_CLI_DESC[DSC$W_LENGTH];   ! and use the file name length
; 2244          2761  3             NAME_DESC[1] = .INFILE_CLI_DESC[DSC$A_POINTER]; ! and length passed by the CLI.
; 2245          2762  2             END;
; 2246          2763  2 !
; 2247          2764  2 ! If mag tape and operator aborted the mount, make it fatal
; 2248          2765  2 !
; 2249          2766  2         IF .FAB_RAB[$FAB_DEV(sdi)]
; 2250          2767  2             AND .FAB_RAB[FAB$L_STV] EQL SS$_ABORT
; 2251          2768  2         THEN
; 2252          2769  2             MESSAGE_ID = MSG$_OPENINX;
; 2253          2770  2
; 2254          2771  2 !
; 2255          2772  2 ! Signal the error condition.
; 2256          2773  2 !
; 2257          2774  2
; 2258        P 2775  2     PUT_MESSAGEX( .MESSAGE_ID,                             ! Signal "input open error" with the following argum
; 2259        P 2776  2             1,                                             !    Number of message arguments
; 2260        P 2777  2             NAME_DESC,                                     !    Address of input name descriptor
; 2261        P 2778  2             .FAB_RAB[FAB$L_STS],                           !    Primary RMS completion code
; 2262          2779  2             .FAB_RAB[FAB$L_STV] );                         !    Secondary RMS completion code
; 2263          2780  2
; 2264          2781  2 !
; 2265          2782  2 ! Return to the caller.
; 2266          2783  2 !
; 2267          2784  2
; 2268          2785  2     RETURN;                                                ! Return to the caller.
; 2269          2786  2
; 2270          2787  1     END;



                             001C 00000          .ENTRY  COPY$INOPN_ERR, Save R2,R3,R4                 ; 2692
          54      0000V  CF    9E 00002          MOVAB   COPY$MSG_NUMBER, R4                           ;
          5E             08    C2 00007          SUBL2   #8, SP                                        ;
          52         04  AC    D0 0000A          MOVL    FAB_RAB_ADDRESS, R2                           ; 2729
          50         28  A2    D0 0000E          MOVL    40(R2), NAM_BLK                               ; 2740
                     03  A0    95 00012          TSTB    3(NAM_BLK)                                    ; 2742
                         10    13 00015          BEQL    1$                                            ;
          53       109A  8F    3C 00017          MOVZWL  #4250, MESSAGE_ID                             ; 2745
          6E         03  A0    9A 0001C          MOVZBL  3(NAM_BLK), NAME_DESC                         ; 2746
     04   AE         04  A0    D0 00020          MOVL    4(NAM_BLK), NAME_DESC+4                       ; 2747
                         25    11 00025          BRB     3$                                            ; 2742
                     0B  A0    95 00027 1$:      TSTB    11(NAM_BLK)                                   ; 2750
                         10    13 0002A          BEQL    2$                                            ;
```

```
                                                L 11
COPYMAIN                                 15-Sep-1984 23:39:26    VAX-11 Bliss-32 V4.0-742               Page  65
V04-000                                  14-Sep-1984 12:14:18    [COPY.SRC]COPYMAIN.B32;1                    (18)

                                   53     109A   8F  3C 0002C      MOVZWL  #4250, MESSAGE_ID                ; 2753
                                   6E       0B   A0  9A 00031      MOVZBL  11(NAM_BLK), NAME_DESC           ; 2754
                              04   AE       0C   A0  D0 00035      MOVL    12(NAM_BLK), NAME_DESC+4         ; 2755
                                                 10  11 0003A      BRB     3$                               ; 2750
                                   53     109C   8F  3C 0003C 2$:  MOVZWL  #4252, MESSAGE_ID                ; 2759
                                   6E    0000G   CF  3C 00041      MOVZWL  INFILE_CLI_DESC, NAME_DESC       ; 2760
                              04   AE    0000G   CF  D0 00046      MOVL    INFILE_CLI_DESC+4, NAME_DESC+4   ; 2761
                        0B    40   A2            04  E1 0004C 3$:  BBC     #4, 64(R2), 4$                   ; 2766
                                   2C       0C   A2  D1 00051      CMPL    12(R2), #44                      ; 2767
                                                 05  12 00055      BNEQ    4$                               ;
                                   53     109C   8F  3C 00057      MOVZWL  #4252, MESSAGE_ID                ; 2769
                                                     53  DD 0005C 4$:  PUSHL   MESSAGE_ID                   ; 2779
                                   64            01  FB 0005E      CALLS   #1, COPY$MSG_NUMBER              ;
            7E          00         50            01  7A 00061      EMUL    #1, R0, #0, -(SP)                ;
            50          50         8E            08  7B 00066      EDIV    #8, (SP)+, R0, R0                ;
                                   04            50  D1 0006B      CMPL    R0, #4                           ;
                                                 18  13 0006E      BEQL    5$                               ;
                                   7E       08   A2  7D 00070      MOVQ    8(R2), -(SP)                     ;
                                            08   AE  9F 00074      PUSHAB  NAME_DESC                        ;
                                                 01  DD 00077      PUSHL   #1                               ;
                                                 53  DD 00079      PUSHL   MESSAGE_ID                       ;
                                   64            01  FB 0007B      CALLS   #1, COPY$MSG_NUMBER              ;
                                                 50  DD 0007E      PUSHL   R0                               ;
                       00000000G   00            05  FB 00080      CALLS   #5, LIB$SIGNAL                   ;
                                                     04 00087      RET                                      ;
                                   7E       08   A2  7D 00088 5$:  MOVQ    8(R2), -(SP)                     ;
                                            08   AE  9F 0008C      PUSHAB  NAME_DESC                        ;
                                                 01  DD 0008F      PUSHL   #1                               ;
                                                 53  DD 00091      PUSHL   MESSAGE_ID                       ;
                                   64            01  FB 00093      CALLS   #1, COPY$MSG_NUMBER              ;
                                                 50  DD 00096      PUSHL   R0                               ;
                       00000000G   00            05  FB 00098      CALLS   #5, LIB$STOP                     ;
                                                     04 0009F      RET                                      ; 2787

; Routine Size:  160 bytes,    Routine Base:  $CODE$ + 09E5
; 2272         2788   1 ROUTINE IN_READ_ERROR : NOVALUE =                 ! RMS input read error action routine
; 2273         2789   1
; 2274         2790   1 !++
; 2275         2791   1 ! FUNCTIONAL DESCRIPTION:
; 2276         2792   1 !
; 2277         2793   1 !       This RMS error action routine sends an input read error message to the user.
; 2278         2794   1 !
; 2279         2795   1 ! FORMAL PARAMETERS:
; 2280         2796   1 !
; 2281         2797   1 !       None
; 2282         2798   1 !
; 2283         2799   1 ! IMPLICIT INPUTS:
; 2284         2800   1 !
; 2285         2801   1 !       INFILE_RAB - Input file RAB
; 2286         2802   1 !       IN_NAME_DESC - Input file name descriptor
; 2287         2803   1 !
; 2288         2804   1 ! IMPLICIT OUTPUTS:
; 2289         2805   1 !
; 2290         2806   1 !       None
; 2291         2807   1 !
; 2292         2808   1 ! ROUTINE VALUE:
; 2293         2809   1 !
; 2294         2810   1 !       None
; 2295         2811   1 !
; 2296         2812   1 ! SIDE EFFECTS:
; 2297         2813   1 !
; 2298         2814   1 !       None
; 2299         2815   1 !
; 2300         2816   1 !--
; 2301         2817   1
; 2302         2818   2     BEGIN
; 2303         2819   2
; 2304         2820   2 !
; 2305         2821   2 ! Signal the input read error.
; 2306         2822   2 !
; 2307         2823   2
; 2308       P 2824   2     PUT_MESSAGE( MSG$_READERR,                    ! Signal a "read error" with the following arguments
; 2309       P 2825   2                  1,                               !    Number of message arguments
; 2310       P 2826   2                  IN_NAME_DESC,                    !    Address of input file name descriptor
; 2311       P 2827   2                  .INFILE_RAB[RAB$L_STS],          !    Primary RMS completion code
; 2312         2828   2                  .INFILE_RAB[RAB$L_STV] );        !    Secondary RMS completion code
; 2313         2829   2
; 2314         2830   2 !
; 2315         2831   2 ! Return to the caller.
; 2316         2832   2 !
; 2317         2833   2
; 2318         2834   2     RETURN;                                       ! Return to the caller.
; 2319         2835   2
; 2320         2836   1     END;



                                 0000 00000 IN_READ_ERROR:
                                                 .WORD   Save nothing                                      ; 2788
                  7E     0000G  CF  7D 00002     MOVQ    INFILE_RAB+8, -(SP)                               ; 2828
```

```
                                   0000G  CF  9F 00007        PUSHAB  IN_NAME_DESC                              :
                                          01  DD 0000B        PUSHL   #1                                        :
                            7E     10B2   8F  3C 0000D        MOVZWL  #4274, -(SP)                              :
                    0000V   CF            01  FB 00012        CALLS   #1, COPY$MSG_NUMBER                       :
                                          50  DD 00017        PUSHL   R0                                        :
               000000000G   00            05  FB 00019        CALLS   #5, LIB$SIGNAL                            :
                                              04 00020        RET                                               : 2836

; Routine Size:  33 bytes,    Routine Base:  $CODE$ + 0A85
```

```
  1 ROUTINE IN_CLOSE_ERROR (                                   ! RMS input close error action routine
  1                 FAB_RAB_ADDRESS )                          ! Address of associated FAB or RAB
  1                 : NOVALUE =
  1
  1 !++
  1 ! FUNCTIONAL DESCRIPTION:
  1 !
  1 !       This RMS error action routine sends an input close error message to the user.
  1 !
  1 ! FORMAL PARAMETERS:
  1 !
  1 !       FAB_RAB_ADDRESS.ra.v - Address of the associated FAB or RAB
  1 !
  1 ! IMPLICIT INPUTS:
  1 !
  1 !       IN_NAME_DESC - Input file name descriptor
  1 !
  1 ! IMPLICIT OUTPUTS:
  1 !
  1 !       None
  1 !
  1 ! ROUTINE VALUE:
  1 !
  1 !       None
  1 !
  1 ! SIDE EFFECTS:
  1 !
  1 !       None
  1 !
  1 !--
  1
  2     BEGIN
  2
  2     BIND
  2         FAB_RAB = .FAB_RAB_ADDRESS :  BLOCK[,BYTE];     ! Redefine routine parameter.
  2
  2 !
  2 ! Signal an input close error.
  2 !
  2
P 2     PUT_MESSAGE( MSG$_CLOSEIN,                             ! Signal a "close error" with the following argument
P 2                 1,                                         !    Number of message arguments
P 2                 IN_NAME_DESC,                              !    Address of input file name descriptor
P 2                 .FAB_RAB[FAB$L_STS],                       !    Primary RMS completion code
  2                 .FAB_RAB[FAB$L_STV] );                     !    Secondary RMS completion code
  2
  2 !
  2 ! Return to the caller.
  2 !
  2
  2     RETURN;                                                ! Return to the caller.
  2
  1     END;
```

```
                                   0000 00000 IN_CLOSE_ERROR:
                                                      .WORD   Save nothing                          ; 2837
                   50       04  AC  D0 00002          MOVL    FAB_RAB_ADDRESS, R0                   ; 2871
                   7E       08  A0  7D 00006          MOVQ    8(R0), -(SP)                          ; 2881
                         0000G  CF  9F 0000A          PUSHAB  IN_NAME_DESC                          ;
                                01  DD 0000E          PUSHL   #1                                    ;
                   7E     1052  8F  3C 00010          MOVZWL  #4178, -(SP)                          ;
              0000V  CF         01  FB 00015          CALLS   #1, COPY$MSG_NUMBER                   ;
                                50  DD 0001A          PUSHL   R0                                    ;
          00000000G  00         05  FB 0001C          CALLS   #5, LIB$SIGNAL                        ;
                                    04 00023          RET                                           ; 2889

; Routine Size:  36 bytes,    Routine Base:  $CODE$ + 0AA6
```

```
1 GLOBAL ROUTINE COPY$OUTOPN_ERR (                               ! RMS output open error action routine
1                 FAB_RAB_ADDRESS )                              ! Address of associated FAB or RAB
1                 : NOVALUE =
1
1 !++
1 ! FUNCTIONAL DESCRIPTION:
1 !
1 !       This RMS error action routine sends an output open error message to the user.
1 !
1 ! FORMAL PARAMETERS:
1 !
1 !       FAB_RAB_ADDRESS.ra.v - Address of the associated FAB or RAB
1 !
1 ! IMPLICIT INPUTS:
1 !
1 !       OUTFILE_NAM_BLK - Output file name block
1 !       OUTFILE_NAME - Output file name after open
1 !       OUTFILE_XNAME - Output file name before open
1 !       OUTFILE_DESC - Output file request descriptor
1 !
1 ! IMPLICIT OUTPUTS:
1 !
1 !       None
1 !
1 ! ROUTINE VALUE:
1 !
1 !       None
1 !
1 ! SIDE EFFECTS:
1 !
1 !       None
1 !
1 !--
1
2     BEGIN
2
2     BIND
2         FAB_RAB = .FAB_RAB_ADDRESS :  BLOCK[,BYTE];     ! Redefine routine parameter.
2
2     LOCAL
2         MESSAGE_ID,                                     ! Local message identifier
2         NAME_DESC :  VECTOR[2];                         ! Output file name descriptor
2
2 !
2 ! Fillin the file name descriptor with the most complete name possible.
2 !
2
2     IF .OUTFILE_NAM_BLK[NAM$B_RSL] NEQ 0                ! If a resultant name string exists,
2     THEN                                                !
3         BEGIN                                           !
3         MESSAGE_ID = MSG$_OPENOUT;                      ! indicate an open error
3         NAME_DESC[0] = .OUTFILE_NAM_BLK[NAM$B_RSL];     ! and fillin the resultant name length
3         NAME_DESC[1] = OUTFILE_NAME;                    ! and address.
3         END
2     ELSE
2         IF .OUTFILE_NAM_BLK[NAM$B_ESL] NEQ 0            ! If RMS created an expanded string but couldn't ope
2         THEN                                            !
```

```
; 2433          2947  3          BEGIN                                      !
; 2434          2948  3          MESSAGE_ID = MSG$_OPENOUT;                 ! indicate an open error
; 2435          2949  3          NAME_DESC[0] = .OUTFILE_NAM_BLK[NAM$B_ESL]; ! and fillin the expanded name length
; 2436          2950  3          NAME_DESC[1] = OUTFILE_XNAME;              ! and address.
; 2437          2951  3          END
; 2438          2952  2      ELSE
; 2439          2953  3          BEGIN
; 2440          2954  3          MESSAGE_ID = MSG$_OPENOUTX;                ! Otherwise, indicate a fatal open error
; 2441          2955  3          NAME_DESC[0] = .OUT_NAME_DESC[ 0 ];        ! and use the file name length
; 2442          2956  3          NAME_DESC[1] = .OUT_NAME_DESC[ 1 ]; ! and length passed by the CLI.
; 2443          2957  3          END;
; 2444          2958  2  !
; 2445          2959  2  ! If mag tape and operator aborted the mount, make it fatal
; 2446          2960  2  !
; 2447          2961  2      IF .FAB_RAB[$FAB_DEV(sdi)]
; 2448          2962  2          AND .FAB_RAB[FAB$L_STV] EQL SS$_ABORT
; 2449          2963  2      THEN
; 2450          2964  2          MESSAGE_ID = MSG$_OPENOUTX;
; 2451          2965  2
; 2452          2966  2  !
; 2453          2967  2  ! Signal the error condition.
; 2454          2968  2  !
; 2455          2969  2
; 2456        P 2970  2    PUT_MESSAGEX( .MESSAGE_ID,                     ! Signal "output open error" with the following argu
; 2457        P 2971  2                1,                                 !    Number of message arguments
; 2458        P 2972  2                NAME_DESC,                         !    Address of output name descriptor
; 2459        P 2973  2                .FAB_RAB[FAB$L_STS],               !    Primary RMS completion code
; 2460          2974  2                .FAB_RAB[FAB$L_STV] );             !    Secondary RMS completion code
; 2461          2975  2
; 2462          2976  2  !
; 2463          2977  2  ! Return to the caller.
; 2464          2978  2  !
; 2465          2979  2
; 2466          2980  2    RETURN;                                        ! Return to the caller.
; 2467          2981  2
; 2468          2982  1    END;



                                 001C 00000          .ENTRY  COPY$OUTOPN_ERR, Save R2,R3,R4                   ; 2890
              54      0000V   CF   9E 00002          MOVAB   COPY$MSG_NUMBER, R4                              ;
              5E              08   C2 00007          SUBL2   #8, SP                                           ;
              52         04   AC   D0 0000A          MOVL    FAB_RAB_ADDRESS, R2                              ; 2927
              50      0000G   CF   9A 0000E          MOVZBL  OUTFILE_NAM_BLK+3, R0                            ; 2937
                              10   13 00013          BEQL    1$                                               ;
              53       10A2   8F   3C 00015          MOVZWL  #4258, MESSAGE_ID                                ; 2940
              6E              50   D0 0001A          MOVL    R0, NAME_DESC                                    ; 2941
       04     AE      0000G   CF   9E 0001D          MOVAB   OUTFILE_NAME, NAME_DESC+4                        ; 2942
                              21   11 00023          BRB     3$                                               ; 2937
              50      0000G   CF   9A 00025  1$:     MOVZBL  OUTFILE_NAM_BLK+11, R0                           ; 2945
                              10   13 0002A          BEQL    2$                                               ;
              53       10A2   8F   3C 0002C          MOVZWL  #4258, MESSAGE_ID                                ; 2948
              6E              50   D0 00031          MOVL    R0, NAME_DESC                                    ; 2949
       04     AE      0000G   CF   9E 00034          MOVAB   OUTFILE_XNAME, NAME_DESC+4                       ; 2950
                              0A   11 0003A          BRB     3$                                               ; 2945
```

```
                                   53      10A4    8F  3C 0003C 2$:      MOVZWL  #4260, MESSAGE_ID                       ; 2954
                                   6E     0000G    CF  7D 00041          MOVQ    OUT_NAME_DESC, NAME_DESC                ; 2955
                      0B       40  A2              04  E1 00046 3$:      BBC     #4, 64(R2), 4$                          ; 2961
                                   2C        0C    A2  D1 0004B          CMPL    12(R2), #44                             ; 2962
                                                   05  12 0004F          BNEQ    4$                                      ;
                                   53      10A4    8F  3C 00051          MOVZWL  #4260, MESSAGE_ID                       ; 2964
                                                   53  DD 00056 4$:      PUSHL   MESSAGE_ID                              ; 2974
                                   64              01  FB 00058          CALLS   #1, COPY$MSG_NUMBER                     ;
          7E                00     50              01  7A 0005B          EMUL    #1, R0, #0, -(SP)                       ;
          50                50     8E              08  7B 00060          EDIV    #8, (SP)+, R0, R0                       ;
                                   04              50  D1 00065          CMPL    R0, #4                                  ;
                                                   18  13 00068          BEQL    5$                                      ;
                                   7E        08    A2  7D 0006A          MOVQ    8(R2), -(SP)                            ;
                                             08    AE  9F 0006E          PUSHAB  NAME_DESC                               ;
                                                   01  DD 00071          PUSHL   #1                                      ;
                                                   53  DD 00073          PUSHL   MESSAGE_ID                              ;
                                   64              01  FB 00075          CALLS   #1, COPY$MSG_NUMBER                     ;
                                                   50  DD 00078          PUSHL   R0                                      ;
                          00000000G  00            05  FB 0007A          CALLS   #5, LIB$SIGNAL                          ;
                                                   04  00081          RET                                             ;
                                   7E        08    A2  7D 00082 5$:      MOVQ    8(R2), -(SP)                            ;
                                             08    AE  9F 00086          PUSHAB  NAME_DESC                               ;
                                                   01  DD 00089          PUSHL   #1                                      ;
                                                   53  DD 0008B          PUSHL   MESSAGE_ID                              ;
                                   64              01  FB 0008D          CALLS   #1, COPY$MSG_NUMBER                     ;
                                                   50  DD 00090          PUSHL   R0                                      ;
                          00000000G  00            05  FB 00092          CALLS   #5, LIB$STOP                            ;
                                                       04 00099          RET                                             ; 2982

; Routine Size:  154 bytes,    Routine Base:  $CODE$ + 0ACA
; 2470          2983  1 ROUTINE OUT_WRITE_ERROR : NOVALUE =                       ! RMS output write error action routine
; 2471          2984  1
; 2472          2985  1 !++
; 2473          2986  1 ! FUNCTIONAL DESCRIPTION:
; 2474          2987  1 !
; 2475          2988  1 !       This RMS error action routine sends an output read error message to the user.
; 2476          2989  1 !
; 2477          2990  1 ! FORMAL PARAMETERS:
; 2478          2991  1 !
; 2479          2992  1 !       None
; 2480          2993  1 !
; 2481          2994  1 ! IMPLICIT INPUTS:
; 2482          2995  1 !
; 2483          2996  1 !       OUTFILE_RAB - Output file RAB
; 2484          2997  1 !       OUT_NAME_DESC - Output file name descriptor
; 2485          2998  1 !
; 2486          2999  1 ! IMPLICIT OUTPUTS:
; 2487          3000  1 !
; 2488          3001  1 !       None
; 2489          3002  1 !
; 2490          3003  1 ! ROUTINE VALUE:
; 2491          3004  1 !
; 2492          3005  1 !       None
; 2493          3006  1 !
; 2494          3007  1 ! SIDE EFFECTS:
; 2495          3008  1 !
; 2496          3009  1 !       None
; 2497          3010  1 !
; 2498          3011  1 !--
; 2499          3012  1
; 2500          3013  2     BEGIN
; 2501          3014  2
; 2502          3015  2 !
; 2503          3016  2 ! Signal the output write error.
; 2504          3017  2 !
; 2505          3018  2
; 2506        P 3019  2     PUT_MESSAGE( MSG$_WRITEERR,                           ! Signal a "write error" with the following argument
; 2507        P 3020  2                  1,                                       !    Number of message arguments
; 2508        P 3021  2                  OUT_NAME_DESC,                           !    Address of output file name descriptor
; 2509        P 3022  2                  .OUTFILE_RAB[RAB$L_STS],                 !    Primary RMS completion code
; 2510          3023  2                  .OUTFILE_RAB[RAB$L_STV] );               !    Secondary RMS completion code
; 2511          3024  2
; 2512          3025  2 !
; 2513          3026  2 ! Return to the caller.
; 2514          3027  2 !
; 2515          3028  2
; 2516          3029  2     RETURN;                                               ! Return to the caller.
; 2517          3030  2
; 2518          3031  1     END;



                                     0000 00000 OUT_WRITE_ERROR:
                                                     .WORD   Save nothing                                              ; 2983
                          7E     0000G  CF   7D 00002        MOVQ    OUTFILE_RAB+8, -(SP)                              ; 3023
```

```
                                  0000G  CF   9F 00007          PUSHAB  OUT_NAME_DESC                              ;
                                              01   DD 0000B     PUSHL   #1                                         ;
                          7E      10D2  8F   3C 0000D           MOVZWL  #4306, -(SP)                               ;
                    0000V CF              01   FB 00012         CALLS   #1, COPY$MSG_NUMBER                        ;
                                          50   DD 00017         PUSHL   R0                                         ;
               000000000G 00              05   FB 00019         CALLS   #5, LIB$SIGNAL                             ;
                                               04 00020         RET                                                ; 3031

; Routine Size:  33 bytes,    Routine Base:  $CODE$ + 0B64
```

```
  1 GLOBAL ROUTINE COPY$OCLOSE_ERR (                              ! RMS output close error action routine
                    FAB_RAB_ADDRESS )                           ! Address of associated FAB or RAB
                    : NOVALUE =
  1
  1 !++
  1 ! FUNCTIONAL DESCRIPTION:
  1 !
  1 !       This RMS error action routine sends an output close error message to the user.
  1 !
  1 ! FORMAL PARAMETERS:
  1 !
  1 !       FAB_RAB_ADDRESS.ra.v - Address of the associated FAB or RAB
  1 !
  1 ! IMPLICIT INPUTS:
  1 !
  1 !       OUT_NAME_DESC - Output file name descriptor
  1 !
  1 ! IMPLICIT OUTPUTS:
  1 !
  1 !       None
  1 !
  1 ! ROUTINE VALUE:
  1 !
  1 !       None
  1 !
  1 ! SIDE EFFECTS:
  1 !
  1 !       None
  1 !
  1 !--
  1
  2     BEGIN
  2
  2     BIND
  2         FAB_RAB = .FAB_RAB_ADDRESS :  BLOCK[,BYTE];           ! Redefine routine parameter.
  2
  2 !
  2 ! Signal an output close error.
  2 !
  2
P 2     PUT_MESSAGE( MSG$_CLOSEOUT,                               ! Signal a "close error" with the following argument
P 2                  1,                                           !    Number of message arguments
P 2                  OUT_NAME_DESC,                               !    Address of output file name descriptor
P 2                  .FAB_RAB[FAB$L_STS],                         !    Primary RMS completion code
  2                  .FAB_RAB[FAB$L_STV] );                       !    Secondary RMS completion code
  2
  2 !
  2 ! Return to the caller.
  2 !
  2
  2     RETURN;                                                   ! Return to the caller.
  2
  1     END;
```

```
                                            0000 00000          .ENTRY  COPY$OCLOSE_ERR, Save nothing                        ; 3032
                     50       04      AC     D0 00002          MOVL    FAB_RAB_ADDRESS, R0                                  ; 3066
                     7E       08      A0     7D 00006          MOVQ    8(R0), -(SP)                                         ; 3076
                           0000G      CF     9F 0000A          PUSHAB  OUT_NAME_DESC                                        ;
                                      01     DD 0000E          PUSHL   #1                                                   ;
                     7E     105A      8F     3C 00010          MOVZWL  #4186, -(SP)                                         ;
              0000V  CF               01     FB 00015          CALLS   #1, COPY$MSG_NUMBER                                  ;
                                      50     DD 0001A          PUSHL   R0                                                   ;
         00000000G   00               05     FB 0001C          CALLS   #5, LIB$SIGNAL                                       ;
                                      04        00023          RET                                                          ; 3084

; Routine Size:  36 bytes,    Routine Base:  $CODE$ + 0B85
```

```
1 GLOBAL ROUTINE COPY$MSG_NUMBER (                                  ! COPY/APPEND message number generator
1                           MSG_ID ) =                             ! Message number
1
1 !++
1 ! FUNCTIONAL DESCRIPTION:
1 !
1 !       This routine return a COPY-specific or APPEND-specific message id
1 !       by inserting the appropriate facility identifier in the high word
1 !       of the message id which is passed by the caller.  This routine also
1 !       records the highest severity message encountered.
1 !
1 ! FORMAL PARAMETERS:
1 !
1 !       MSG_ID.rlu.v - Message id
1 !
1 ! IMPLICIT INPUTS:
1 !
1 !       APPEND_COMMAND = APPEND command indicator
1 !       MOST_SEVERE_ERR - Current most severe error id
1 !       OUTFILE_NAM_BLK - Output file name block - wildcard indicator
1 !
1 ! IMPLICIT OUTPUTS:
1 !
1 !       MOST_SEVERE_ERR - Most severe error id may be updated
1 !
1 ! ROUTINE VALUE:
1 !
1 !       Actual message id
1 !
1 ! SIDE EFFECTS:
1 !
1 !       None
1 !
1 !--
1
2     BEGIN
2
2     MAP                                                           !
2         MSG_ID :  BLOCK[,BYTE];                                   ! Redefine the form of the input argument
2
2     LOCAL
2         ACTUAL_MSG_ID :  BLOCK[1];                                ! Actual message identifier
2
2 !
2 ! Calculate the actual message identifier.
2 !
2
2 IF .MSG_ID<16,16> EQL 0                                          ! If facility unspecified.
2 THEN
2     IF .APPEND_COMMAND                                            ! If this is an APPEND command,
2     THEN                                                          !
3         ACTUAL_MSG_ID = .MSG_ID + (APPEND_ID * 65536)             ! insert the APPEND facility code into the message i
2     ELSE                                                          ! If this is a COPY command,
3         ACTUAL_MSG_ID = .MSG_ID + (COPY_ID * 65536)               ! insert the COPY facility code into the message id.
2 ELSE
2     ACTUAL_MSG_ID = .MSG_ID;                                      ! else use existing code
2
```

```
; 2631          3142  2 !
; 2632          3143  2 ! Update the "most severe error" if the current error is more severe.
; 2633          3144  2 !
; 2634          3145  2
; 2635          3146  2     IF NOT .ACTUAL_MSG_ID AND                          ! If the current message is not a success message an
; 2636          3147  3         (.MOST_SEVERE_ERR OR                           ! either this is the first error message
; 2637          3148  3         .ACTUAL_MSG_ID[STS$V_SEVERITY] GTRU            ! or the current message severity
; 2638          3149  3               .MOST_SEVERE_ERR[STS$V_SEVERITY])        ! is greater than the previous severity,
; 2639          3150  2     THEN                                               !
; 2640          3151  2         MOST_SEVERE_ERR = .ACTUAL_MSG_ID OR            ! update the most severe message id
; 2641          3152  2                               STS$M_INHIB_MSG;        ! and turn on the "suppress message" indicator.
; 2642          3153  2
; 2643          3154  2 !
; 2644          3155  2 ! Return the actual message id to the caller.
; 2645          3156  2 !
; 2646          3157  2
; 2647          3158  2     RETURN .ACTUAL_MSG_ID;                             ! Return the actual message id to the caller.
; 2648          3159  2
; 2649          3160  1     END;



                                 0004 00000         .ENTRY  COPY$MSG_NUMBER, Save R2                        ; 3085
                 52     0000'  CF   9E 00002         MOVAB   MOST_SEVERE_ERR, R2                             ;
                           06  AC   B5 00007         TSTW    MSG_ID+2                                        ; 3132
                               1A   12 0000A         BNEQ    2$                                              ;
                 0B        10  A2   E9 0000C         BLBC    COPY$CLI_STATUS, 1$                             ; 3134
       50        04  AC 00710000  8F   C1 00010      ADDL3   #7405568, MSG_ID, ACTUAL_MSG_ID                 ; 3136
                               0F   11 00019         BRB     3$                                              ;
       50        04  AC 00670000  8F   C1 0001B 1$:  ADDL3   #6750208, MSG_ID, ACTUAL_MSG_ID                 ; 3138
                               04   11 00024         BRB     3$                                              ; 3134
                 50        04  AC   D0 00026 2$:     MOVL    MSG_ID, ACTUAL_MSG_ID                           ; 3140
                 17            50   E8 0002A 3$:     BLBS    ACTUAL_MSG_ID, 5$                               ; 3146
                 0C            62   E8 0002D         BLBS    MOST_SEVERE_ERR, 4$                             ; 3147
  51       62    03            00   EF 00030         EXTZV   #0, #3, MOST_SEVERE_ERR, R1                     ; 3149
  51       50    03            00   ED 00035         CMPZV   #0, #3, ACTUAL_MSG_ID, R1                       ;
                               08   1B 0003A         BLEQU   5$                                              ;
       62        50 10000000  8F   C9 0003C 4$:      BISL3   #268435456, ACTUAL_MSG_ID, MOST_SEVERE_ERR      ; 3151
                               04 00044 5$:          RET                                                     ; 3160

; Routine Size:  69 bytes,    Routine Base:  $CODE$ + 0BA9
; 2651          3161  1 END
; 2652          3162  0 ELUDOM



                                                                            .EXTRN  LIB$SIGNAL, LIB$STOP
;                                          PSECT SUMMARY
;
;       Name                           Bytes                        Attributes
;
;    $GLOBAL$                             61  NOVEC,  WRT,  RD ,NOEXE,NOSHR,  LCL,  REL,  CON,NOPIC,ALIGN(2)
;    $PLIT$                               92  NOVEC,NOWRT,  RD ,NOEXE,NOSHR,  LCL,  REL,  CON,NOPIC,ALIGN(2)
;    $CODE$                             3054  NOVEC,NOWRT,  RD ,  EXE,NOSHR,  LCL,  REL,  CON,NOPIC,ALIGN(2)
;    COPY$COPY_FILE                      180  NOVEC,NOWRT,  RD ,  EXE,NOSHR,  LCL,  REL,  CON,NOPIC,ALIGN(9)
;    $OWN$                                 4  NOVEC,  WRT,  RD ,NOEXE,NOSHR,  LCL,  REL,  CON,NOPIC,ALIGN(2)


;                              Library Statistics
;
;                                              -------- Symbols --------       Pages       Processing
;       File                                   Total   Loaded   Percent       Mapped      Time
;
;    _$255$DUA28:[SYSLIB]STARLET.L32;1         9776      150         1         581          00:01.1



;                                          COMMAND QUALIFIERS

;       BLISS/CHECK=(FIELD,INITIAL,OPTIMIZE)/LIS=LIS$:COPYMAIN/OBJ=OBJ$:COPYMAIN MSRC$:COPYMAIN/UPDATE=(ENH$:COPYMAIN)

; Size:           3234 code + 157 data bytes
; Run Time:          01:05.8
; Elapsed Time:      02:27.9
; Lines/CPU Min:       2883
; Lexemes/CPU-Min:  23243
; Memory Used:   277 pages
; Compilation Complete
```

0067 AH-BT13A-SE
VAX/VMS V4.0

DIGITAL EQUIPMENT CORPORATION
CONFIDENTIAL AND PROPRIETARY